Consequence-Based and Fixed-Parameter Tractable Reasoning in Description Logics

František Simančík and Boris Motik and Ian Horrocks

Department of Computer Science, University of Oxford, UK

Abstract

In this paper we investigate the consequence-based algorithms that are nowadays commonly used for subsumption reasoning with description logic ontologies, presenting the following novel results. First, we present a very general consequence-based reasoning algorithm that can be instantiated so as to capture the essential features of the known algorithms. Second, we use this algorithm to develop a novel framework for a quantitative and parametric analysis of the complexity of subsumption reasoning in description logic ontologies. Our approach is based on the notion of a *decomposition*—a graph-like structure that, roughly speaking, summarizes the models relevant during ontology reasoning. We identify *width* and *length* as decomposition parameters that determine the "level" of combinatorial reasoning. We prove that, when parameterized by a decomposition, our consequence-based algorithm runs in time that is fixed-parameter tractable in width and length. Third, we briefly discuss how length and width characterize the behavior of tableau algorithms. Fourth, we show that the width and length of existing ontologies are reasonably small, which, we believe, explains the good performance of consequence-based algorithms in practice. We thus obtain a sound foundation for a practical implementation of ontology reasoners, as well as a way to analyze the reasoners' performance.

Keywords: description logics, parameterized complexity, treewidth

1. Introduction

An *ontology* describes an application domain in terms of *concepts* (i.e., the kinds of objects in the domain), *roles* (i.e., the relationships between objects), and *axioms* (i.e., statements that formally describe concepts and roles). Axioms are logical statements and can thus be automatically processed via *reasoning*. For example, an ontology describing the structure of a university might use concepts *Employee*, *Professor*, and *AssistantProfessor* to represent employees, professors, and assistant professors, respectively, and the university's organizational hierarchy might be captured by axioms stating that *Professor* is a subconcept of *Employee*, and that *AssistantProfessor* is a subconcept of *Professor*. An *ontology reasoner* can use such statements to conclude that *AssistantProfessor* is also a subconcept of *Employee*. This is an example of *subsumption reasoning* (i.e., reasoning about the hierarchy of ontology concepts), which is a central computational problem in ontology-based applications in areas as diverse as biology [1], medicine [2], geography [3], astronomy [4], agriculture [5], and defense [6].

The Web Ontology Language (OWL) [7] and its successor OWL 2 [8, 9] are standard ontology languages developed by the World Wide Web Consortium (W3C). Their logical underpinning is provided by *description logics* (DLs) [10]—a well-known family of knowledge representation languages. Reasoning with expressive DLs can be quite complex, so for simplicity in this paper we focus on the basic description logic $\mathcal{ALCI}$ that provides the fundamental constructs such as inverse roles, full Boolean connectives on concepts, and existential and universal quantification; however, quantification is *guarded* [11], which ensures that subsumption reasoning is decidable, albeit of high complexity (ExpTime-complete). The reason for this high complexity is the size and number of relevant models [10, Ch. 3]: due to an interaction between existential and universal quantifiers, an ontology may be satisfied only in models containing an exponential number of objects (an effect known as *and-branching*), and due to disjunctions, an ontology may admit an exponential number of "relevant" models (an effect known as *or-branching*). State of the art OWL reasoners, including Pellet [12], FaCT++ [13], and HermiT [14], are all based on variants of the tableau algorithm, which try to construct a finite representation of a model of an ontology. The basic algorithm has been heavily optimized so as to curb and- and or-branching in practice [10, Ch. 9], which allows DL reasoners to successfully process many large and nontrivial ontologies. The performance of such reasoners is, however, relatively brittle, and there exist ontologies (for example the GALEN ontology[1]) that none of these reasoners is able to process. Although there has been some work on identifying the *qualitative* features of an ontology that may degrade the performance of a given reasoner [15], no existing method can provide a *quantitative* measure of the "difficulty" of reasoning with a particular ontology.

In order to provide more robust performance of reasoning, one can reduce language expressivity so as to make reasoning easier. By removing disjunctions one obtains the Horn family of DLs [16]. Horn ontologies can always be translated into

[1] `http://opengalen.org/`

Horn clauses, which eliminates all or-branching; however, while this has been empirically shown to make reasoning easier [14], reasoning with Horn-$\mathcal{ALCI}$ is still ExpTime-hard due to and-branching [17]. By further removing universal quantification and inverse roles from Horn-$\mathcal{ALCI}$ one obtains the $\mathcal{EL}$ family of DLs [18], for which subsumption reasoning is tractable; similarly, by removing universal quantification, qualified existential quantification, and conjunction from Horn-$\mathcal{ALCI}$ one obtains the DL-Lite family of DLs [19, 20], for which subsumption reasoning is also tractable. Although Horn DLs have been successfully applied in practice, many applications require inverse roles (which precludes the use of $\mathcal{EL}$), qualified existential quantification (which precludes the use of DL-Lite), or concept covering constraints (which requires disjunction and hence precludes the use of Horn DLs altogether). Thus, improving reasoning performance by reducing expressivity is not always feasible.

An alternative is to explore new reasoning techniques that may be better adapted to practical ontologies. *Consequence-based* algorithms [21, 22] have recently proved to be very effective in practice; for example, the CB reasoner is currently the only reasoner that can process the GALEN ontology in its entirety [21], and the ELK reasoner has revolutionized the processing of large $\mathcal{EL}$ ontologies such as SNOMED CT[2] [23]. Intuitively, such algorithms can be seen as variants of resolution optimized to reduce the number of clauses produced on typical ontologies. However, although consequence-based reasoners have been very successful, it is still not possible to quantify the difficulty of reasoning with a specific ontology.

We believe that methods for the quantitative analysis of the difficulty of reasoning with a specific ontology $\mathcal{O}$ would have major benefits in practice. On the one hand, users would be able to predict the effectiveness of reasoners on $\mathcal{O}$, and analyze performance problems if they did arise; and on the other hand, algorithm designers would gain a deeper understanding as to why reasoning with practical ontologies is often easy notwithstanding the high worst case complexity of the problem, and could exploit this understanding in order to design more robustly scalable reasoning algorithms. Crucially, this analysis must go beyond worst-case complexity and consider the salient features of a specific ontology. One can frame this problem in terms of *parameterized complexity* [24], which measures the difficulty of a computational problem not only w.r.t. the *size n* of a problem instance, but also a *parameter k* that quantifies specific aspects of the instance. Of particular interest are *fixed-parameter tractable* (FPT) problems, which can be solved in time $f(k) \cdot n^c$ for a fixed constant c and a fixed computable function f; such problems are interesting because one can hope to solve large instances provided that the parameter remains "small." For example, many hard graph-theoretic problems (including graph homomorphism and 3-colorability) are FPT when the parameter is the graph's *treewidth* [25]—a measure of the graph's similarity to a tree. Furthermore, the notion of treewidth has also been extended to propositional formulae and has been used to obtain FPT results for propositional satisfiability [26]. A key question in parameterized complexity is to identify natural parameters that accurately characterize the problem's difficulty.

In this paper, we present what we believe to be the first framework for a quantitative, parameterized analysis of the complexity of subsumption reasoning with $\mathcal{ALCI}$. Via well-known reductions of role inclusions and role transitivity axioms, our approach can also be applied to $\mathcal{SHI}$ ontologies, and so it covers a significant subset of OWL 2 DL (the latter further allows constructs such as nominals and number restrictions). We characterize the difficulty of subsumption reasoning in DLs using a graph-like structure called a *decomposition*. Furthermore, we present a family of reasoning algorithms based on decompositions that facilitate FPT reasoning. Finally, we show how the good performance of consequence-based reasoners on several widely used ontologies can be explained by an analysis of their decompositions. We next summarize the main contributions of our paper.

In Section 3 we introduce a general consequence-based framework for subsumption reasoning in $\mathcal{ALCI}$ and $\mathcal{SHI}$, and then we discuss possibilities for instantiating the framework using different initialization and expansion strategies. Our algorithms are not only refutationally complete: they can actively derive queries (i.e., axioms of a specific form) that logically follow from a given ontology. A specific instantiation of our framework captures the algorithm used in CB [21] and is closely related to the algorithm used in ConDOR [22]. As well as capturing existing consequence-based approaches, our framework incorporates redundancy deletion and optimized inference rules that have the potential to further improve the practical performance of consequence-based algorithms. Thus, we see our framework as an important contribution in its own right.

In Section 4 we introduce the central notion of a *decomposition* $\mathcal{D}$ of an ontology $\mathcal{O}$ and a set of queries $\mathcal{Q}$. Roughly speaking, $\mathcal{D}$ is a graph-like structure that "summarizes" the models of $\mathcal{O}$ relevant for answering the queries in $\mathcal{Q}$; each vertex of $\mathcal{D}$ identifies a propositional subproblem, and each edge of $\mathcal{D}$ identifies a pathway for the exchange of information between such subproblems. We then show how to instantiate our consequence-based framework from Section 3 so that it can be applied to $\mathcal{D}$, and we identify the *length* and *width* of $\mathcal{D}$ as parameters that characterize the and- and or-branching encountered in this process. Finally, we show that the algorithm can be implemented so that it is fixed-parameter tractable w.r.t. decomposition length and width.

In Section 4.3 we show that decompositions also explain to an extent the difficulty of tableau reasoning. In particular, decomposition width bounds the size of sets of concepts annotating tableau vertices, so length and width bound the size of the constructed model representations. We also point out, however, that tableau algorithms are not FPT because the way in which they construct model representations may result in redundant computations.

In Section 4.4 we present an algorithm for computing a decomposition of $\mathcal{O}$ and $\mathcal{Q}$. One would naturally want to compute a decomposition of smallest length and width; however, in Section 7 we show that there is a tension between these two parameters

[2] http://ihtsdo.org/

by exhibiting a family of ontologies for which each decomposition of minimum width has exponential length. To address this problem, our decomposition construction algorithm is parameterized by a collection of parameters called a *control* that imposes an upper bound on decomposition length so that a unique decomposition of $\mathcal{O}$ and $\mathcal{Q}$ can be computed in polynomial time.

In Section 5 we further extend our results using ideas from reasoning with propositional problems of bounded treewidth. More specifically, we present the notion of *ϵ-refinement* of a decomposition $\mathcal{D}$ which, roughly speaking, is obtained by replacing each vertex of $\mathcal{D}$ with a tree decomposition of the propositional problem corresponding to the vertex. This can reduce the width of a decomposition while increasing the length only by a linear factor, so it can reduce the complexity of reasoning. As for tree decompositions, computing an ϵ-refinement of bounded width (if one exists) is FPT.

In Section 6 we present several soundness- and admissibility-preserving transformations for decompositions that can be used to eliminate redundant information. In Section 7 we then establish bounds on the sizes of the decompositions of $\mathcal{O}$ and $\mathcal{Q}$. For the lower bound, we show that ontologies exist for which all decompositions of minimal width have exponential length; for the upper bound, we use the transformations from Section 6 to show that $\mathcal{O}$ and $\mathcal{Q}$ always admit a decomposition of minimal width with at most exponential length.

In Section 8 we investigate the width and length of several ontologies commonly used in practice. Our results show that almost all ontologies admit a decomposition of length about twice the number of concepts in the ontology, and of width below 30. These results, we believe, explain the good performance of consequence-based reasoning algorithms in practice.

To the best of our knowledge, these are the first results on quantitative analysis and fixed-parameter complexity of ontology reasoning. The only related work we are aware of is the analysis of conditions under which query answering for a given ontology (independently of the DL language) becomes tractable w.r.t. data complexity [27]; however, these results are not quantitative and they do not address the problem of subsumption reasoning. Thus, we believe that this paper provides a completely new perspective on the problem of identifying the sources of complexity in ontology reasoning. Moreover, our results are amenable to practical implementation, and we believe they can significantly improve the scalability and robustness of reasoners.

2. Preliminaries

In this section we recapitulate some well-known definitions and introduce the notation that we use in the rest of our paper.

2.1. Description Logics

Description logics $\mathcal{ALCI}$ and $\mathcal{SHI}$ are defined w.r.t. a fixed *signature* $\Sigma = \langle \Sigma_A, \Sigma_T \rangle$, where Σ_A and Σ_T are disjoint sets of *atomic concepts* and *atomic roles*, respectively.

An *inverse role* is an expression T^- for T an atomic role. A *role* is an atomic role or an inverse role, and Σ_R is the set of all roles. For an atomic role T, let $\mathsf{inv}(T) := T^-$ and $\mathsf{inv}(T^-) := T$.

The set of *concepts* is the smallest set that contains $\top$ (*the top concept*), $\bot$ (*the bottom concept*), each atomic concept A, and $\neg D$ (*negation*), $D_1 \sqcap D_2$ (*conjunction*), $D_1 \sqcup D_2$ (*disjunction*), $\exists R.D$ (*existential restriction*), and $\forall R.D$ (*universal restriction*) for each role R and all concepts D, D_1, and D_2. We assume that $\{\top, \bot\} \cap \Sigma_A = \emptyset$—that is, $\top$ and $\bot$ are not atomic concepts. A *literal* is a concept of the form A, $\exists R.A$, or $\forall R.A$, for A an atomic concept and R a role, and Σ_L is the set of all literals. For $\mathbf{L}$ a set of literals, $\mathbf{L}^\exists$ is the subset of $\mathbf{L}$ containing all literals of the form $\exists R.A$.

Unless otherwise stated, certain letters of the alphabet (possibly with sub- and/or superscripts) are used consistently throughout this paper as specified below, so we do not qualify these letters whenever no ambiguity can arise:

- letters A, B, and C denote atomic concepts,
- letter D denotes a concept,
- letter L denotes a literal,
- letter T denotes an atomic role,
- letters R and S denote roles,
- letter K denotes a conjunction of literals, and
- letter M denotes a disjunction of literals.

We identify conjunctions and disjunctions of literals with sets of literals (i.e., conjunctions and disjunctions of literals are unordered and without repeated literals) and we use them in standard set operations; furthermore, we identify the empty conjunction and the empty disjunction with $\top$ and $\bot$, respectively. Conversely, for U a set of literals, $\bigsqcap U$ and $\bigsqcup U$ are the conjunction and the disjunction, respectively, of the literals in U; furthermore, to simplify the notation, in some cases we will explicitly note that we abbreviate $\bigsqcap U$ as just U. The cardinality of a set N is written $|N|$.

An *axiom* is an expression of the form $D_1 \sqsubseteq D_2$ (*general concept inclusion*), $R_1 \sqsubseteq R_2$ (*role inclusion*), or $\mathsf{Tra}(R)$ (*role transitivity axiom*). A $\mathcal{SHI}$ *ontology* $\mathcal{O}$ is a set of axioms. In this paper we consider only terminological reasoning, so we do not allow for facts (aka ABoxes) in ontologies. The *size* of $\mathcal{O}$, written $\|\mathcal{O}\|$, is the number of symbols in $\mathcal{O}$.

Table 1: The Semantics of $\mathcal{SHI}$

Interpretation of Roles and Concepts	
$(T^{-})^{\mathcal{I}}$	$= \{\langle \delta_2, \delta_1 \rangle \in \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \mid \langle \delta_1, \delta_2 \rangle \in T^{\mathcal{I}}\}$
$\top^{\mathcal{I}}$	$= \Delta^{\mathcal{I}}$
$\bot^{\mathcal{I}}$	$= \emptyset$
$(\neg D)^{\mathcal{I}}$	$= \Delta^{\mathcal{I}} \setminus D^{\mathcal{I}}$
$(D_1 \sqcap D_2)^{\mathcal{I}}$	$= D_1^{\mathcal{I}} \cap D_2^{\mathcal{I}}$
$(D_1 \sqcup D_2)^{\mathcal{I}}$	$= D_1^{\mathcal{I}} \cup D_2^{\mathcal{I}}$
$(\exists R.D)^{\mathcal{I}}$	$= \{\delta_1 \in \Delta^{\mathcal{I}} \mid \exists \delta_2.\langle \delta_1, \delta_2 \rangle \in R^{\mathcal{I}} \wedge \delta_2 \in D^{\mathcal{I}}\}$
$(\forall R.D)^{\mathcal{I}}$	$= \{\delta_1 \in \Delta^{\mathcal{I}} \mid \forall \delta_2.\langle \delta_1, \delta_2 \rangle \in R^{\mathcal{I}} \rightarrow \delta_2 \in D^{\mathcal{I}}\}$

Satisfaction of Axioms in an Interpretation	
$\mathcal{I} \models D_1 \sqsubseteq D_2$	if $(D_1)^{\mathcal{I}} \subseteq (D_2)^{\mathcal{I}}$
$\mathcal{I} \models R_1 \sqsubseteq R_2$	if $(R_1)^{\mathcal{I}} \subseteq (R_2)^{\mathcal{I}}$
$\mathcal{I} \models \mathsf{Tra}(R)$	if $\forall \delta_1 \forall \delta_2 \forall \delta_3.\langle \delta_1, \delta_2 \rangle \in R^{\mathcal{I}} \wedge \langle \delta_2, \delta_3 \rangle \in R^{\mathcal{I}} \rightarrow \langle \delta_1, \delta_3 \rangle \in R^{\mathcal{I}}$

Ontologies are interpreted using Tarski-style semantics. An *interpretation* $\mathcal{I} = \langle \Delta^{\mathcal{I}}, \cdot^{\mathcal{I}} \rangle$ is a pair where $\Delta^{\mathcal{I}}$ is a nonempty set called the *domain*, and $\cdot^{\mathcal{I}}$ is a function that assigns a set $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$ to each atomic concept $A \in \Sigma_A$, and a set $T^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ to each atomic role $T \in \Sigma_T$. Function $\cdot^{\mathcal{I}}$ is extended to roles and concepts as shown in the upper part of Table 1. The satisfaction of an axiom α in $\mathcal{I}$, written $\mathcal{I} \models \alpha$, is defined as shown in the lower part of Table 1. Interpretation $\mathcal{I}$ is a *model* of an ontology $\mathcal{O}$, written $\mathcal{I} \models \mathcal{O}$, if $\mathcal{I}$ satisfies all axioms in $\mathcal{O}$. An ontology is $\mathcal{O}$ *satisfiable* if a model of $\mathcal{O}$ exists; furthermore, $\mathcal{O}$ *entails* an axiom α, written $\mathcal{O} \models \alpha$, if $\mathcal{I} \models \alpha$ for each model $\mathcal{I}$ of $\mathcal{O}$. Note that $\mathcal{O}$ is unsatisfiable if and only if $\mathcal{O} \models \top \sqsubseteq \bot$.

A *clause* is a general concept inclusion of the form $\bigsqcap_{i=1}^{m} L_i \sqsubseteq \bigsqcup_{i=m+1}^{n} L_i$ where $0 \leq m \leq n$ and each L_i is a literal; the clause is *normal* if each L_i with $1 \leq i \leq m$ is an atomic concept; and the clause is a *query* if each L_i with $m + 1 \leq i \leq n$ is an atomic concept. Conjunction K in a clause $K \sqsubseteq M$ is the *antecedent*, and disjunction M is the *consequent*. An ontology $\mathcal{O}$ is *normalized* if each general concept inclusion in $\mathcal{O}$ is a normal clause. Each $\mathcal{SHI}$ ontology $\mathcal{O}$ can be transformed in linear time to a normalized $\mathcal{SHI}$ ontology $\mathcal{O}'$ such that $\mathcal{O}'$ is a conservative extension of $\mathcal{O}$. Several such transformations are well known (see, e.g., [21, 14]), so we omit the details for brevity. Given a set of literals $\mathbf{L}$, a clause $K \sqsubseteq M$ is *over* $\mathbf{L}$ if $K \cup M \subseteq \mathbf{L}$. A clause $K' \sqsubseteq M'$ is a *strengthening* of a clause $K \sqsubseteq M$ if $K' \subseteq K$ and $M' \subseteq M$;[3] furthermore, $K \sqsubseteq M \mathrel{\hat{\in}} \mathcal{N}$ means that a set of clauses $\mathcal{N}$ contains at least one strengthening of $K \sqsubseteq M$.

We consider several fragments of $\mathcal{SHI}$, which, for simplicity, we define only for normalized ontologies. Let $\mathcal{O}$ be a normalized $\mathcal{SHI}$ ontology. Due to normalization, $\mathcal{O}$ cannot contain axioms of the form $\exists R.A \sqcap K \sqsubseteq M$, so the following definitions of various fragments are not standard; however, it is straightforward to see that they correspond to the conventional definitions used in the literature. Furthermore, note that $\exists R.\top$ is not a literal since we do not consider $\top$ to be an atomic concept; however, one can always replace $\top$ with a fresh atomic concept $A_\top$ and axiomatize the latter using a clause $\top \sqsubseteq A_\top$. Now $\mathcal{O}$ is *Horn* if the consequent of each clause in $\mathcal{O}$ contains at most one literal; $\mathcal{O}$ is an $\mathcal{ALCI}$ ontology if each axiom in $\mathcal{O}$ is a general concept inclusion; $\mathcal{O}$ is an $\mathcal{EL}$ ontology if $\mathcal{O}$ is a Horn-$\mathcal{ALCI}$ ontology, each role occurring in $\mathcal{O}$ in an existential restriction is an atomic role, and each role occurring in $\mathcal{O}$ in a universal restriction is an inverse role; and $\mathcal{O}$ is a DL-Lite$_{horn}$ ontology if $\mathcal{O}$ is a Horn-$\mathcal{ALCI}$ ontology and universal restrictions occur in $\mathcal{O}$ only in axioms of the form $\top \sqsubseteq \forall R.B$.

A normalized $\mathcal{SHI}$ ontology $\mathcal{O}$ can be transformed into a normalized $\mathcal{ALCI}$ ontology by eliminating all role inclusions and role transitivity axioms. Towards this goal, let $\sqsubseteq_{\mathcal{O}}$ be the smallest reflexive and transitive binary relation on roles such that $R \sqsubseteq_{\mathcal{O}} S$ and $\mathsf{inv}(R) \sqsubseteq_{\mathcal{O}} \mathsf{inv}(S)$ for each role inclusion $R \sqsubseteq S \in \mathcal{O}$; furthermore, for each role R and each atomic concept C occurring in $\mathcal{O}$, let $A_{R,C}$ and $B_{R,C}$ be fresh atomic concepts unique for R and C. The elimination proceeds in three steps: step 1 ensures that universal restrictions occur only in clauses of the form $A \sqsubseteq \forall R.C$, step 2 encodes transitivity axioms using the well-known "box pushing" method, and step 3 eliminates role inclusions by expanding the role hierarchy into universal restrictions. We capture these steps by defining three ontologies obtained by transforming $\mathcal{O}$ as follows.

1. For each clause $K \sqsubseteq M \in \mathcal{O}$, ontology $\mathcal{O}_1$ contains the clause obtained from $K \sqsubseteq M$ by replacing each literal $\forall R.C \in M$ with $A_{R,C}$; furthermore, for each literal $\forall R.C$ occurring in a clause in $\mathcal{O}$, ontology $\mathcal{O}_1$ contains the axiom $A_{R,C} \sqsubseteq \forall R.C$.

[3] Note that, by this definition, each clause is a strengthening of itself.

Table 2: Derivation Rules of the Hypertableau Algorithm

Hyp-rule	If $A_1 \sqcap \ldots \sqcap A_m \sqsubseteq L_1 \sqcup \ldots \sqcup L_n \in \mathcal{O}$, and an individual $s \in \mathsf{ind}(\mathcal{A}_0)$ exists that is not indirectly blocked such that $A_i(s) \in \mathcal{A}_0$ for each $1 \leq i \leq m$ and $L_j(s) \notin \mathcal{A}_0$ for each $1 \leq j \leq n$, then $\mathcal{A}_1 := \mathcal{A}_0 \cup \{\bot\}$ if $n = 0$; and $\mathcal{A}_j := \mathcal{A}_0 \cup \{L_j(s)\}$ for $1 \leq j \leq n$ if $n > 0$.
∃-rule	If an individual s exists that is not blocked such that $\exists R.A(s) \in \mathcal{A}_0$ and no individual t exists such that $\{R(s,t),\ A(t)\} \subseteq \mathcal{A}_0$, then $\mathcal{A}_1 := \mathcal{A}_0 \cup \{R(s,u),\ A(u)\}$ for u a fresh successor of s.
$\forall^+$-rule	If individuals s and t exists that are not indirectly blocked such that $\{\forall R.B(s),\ R(s,t)\} \subseteq \mathcal{A}_0$ and $B(t) \notin \mathcal{A}_0$, then $\mathcal{A}_1 := \mathcal{A}_0 \cup \{B(t)\}$.
$\forall^-$-rule	If individuals s and t exists that are not indirectly blocked such that $\{[\forall \mathsf{inv}(R).B](t),\ R(s,t)\} \subseteq \mathcal{A}_0$ and $B(s) \notin \mathcal{A}_0$, then $\mathcal{A}_1 := \mathcal{A}_0 \cup \{B(s)\}$.

2. Ontology $\mathcal{O}_2$ contains each clause in $\mathcal{O}_1$; furthermore, for each clause $A \sqsubseteq \forall R.C \in \mathcal{O}_1$ and each role S such that $S \sqsubseteq_{\mathcal{O}} R$ and either $\mathsf{Tra}(S) \in \mathcal{O}$ or $\mathsf{Tra}(\mathsf{inv}(S)) \in \mathcal{O}$, ontology $\mathcal{O}_2$ contains clauses

$$A \sqsubseteq \forall S.B_{S,C} \qquad \text{and} \qquad B_{S,C} \sqsubseteq \forall S.B_{S,C} \qquad \text{and} \qquad B_{S,C} \sqsubseteq C.$$

3. Ontology $\mathcal{O}_3$ contains each clause in $\mathcal{O}_2$; furthermore, for each clause $A \sqsubseteq \forall R.C \in \mathcal{O}_2$ and each role S such that $S \sqsubseteq_{\mathcal{O}} R$, ontology $\mathcal{O}_3$ contains clause $A \sqsubseteq \forall S.C$.

For each clause $K \sqsubseteq M$ where K and M contain only atomic concepts, $\mathcal{O} \models K \sqsubseteq M$ if and only if $\mathcal{O}_3 \models K \sqsubseteq M$; this can be proved using a straightforward adaptation of the proofs by Demri and de Nivelle [28].

The algorithms presented in this paper take a normalized $\mathcal{ALCI}$ ontology $\mathcal{O}$ and a set of queries $\mathcal{Q}$, and they compute the status of $\mathcal{O} \models K \sqsubseteq M$ for each query $K \sqsubseteq M \in \mathcal{Q}$; hence, our algorithms compute functions rather than solving decision problems. For our complexity results, however, we consider a related decision problem of determining whether $\mathcal{O} \models K \sqsubseteq M$ holds for each query $K \sqsubseteq M \in \mathcal{Q}$—that is, the decision problem checks the conjunction of all such entailments.

2.2. The Hypertableau Algorithm

Most OWL 2 DL reasoners are currently based on variants of tableau algorithms. To relate our techniques to the state of the art in ontology reasoning, we next present a variant of the hypertableau algorithm by Motik et al. [14].

We assume the existence of a countably infinite set of *named individuals*. An *individual* is a finite string of the form $a.i_1.\ldots.i_n$ where a is a named individual and $i_1.\ldots.i_n$ is a possibly empty sequence of integers. An individual of the form $s.i$ is *unnamed*, and it is a *successor* of individual s; *predecessor* is the inverse of successor; and *ancestor* and *descendant* are transitive closures of predecessor and successor, respectively. An *ABox* $\mathcal{A}$ is a finite set of facts of the form $\bot$, $L(s)$, or $R(s,t)$, for L a literal, R a role, and s and t individuals; furthermore, $\mathsf{ind}(\mathcal{A})$ is the set of individuals occurring in $\mathcal{A}$.

Termination of the algorithm is ensured via *anywhere blocking*; for $\mathcal{ALCI}$, the *single anywhere* variant suffices. Let $\lhd$ be an arbitrary strict order (i.e., an irreflexive and transitive relation) on the set of all individuals compatible with the ancestor relation (i.e., $s \lhd t$ holds for all individuals s and t such that s is an ancestor of t). The *label* of an individual s in an ABox $\mathcal{A}$ is defined as $\mathcal{L}_{\mathcal{A}}(s) = \{L \mid L(s) \in \mathcal{A}\}$. By induction on $\lhd$, each individual s occurring in $\mathcal{A}$ is assigned a status as follows:

- s is *directly blocked* by an individual s' in $\mathcal{A}$ if both s and s' are unnamed, no ancestor of s' is blocked, $\mathcal{L}_{\mathcal{A}}(s) = \mathcal{L}_{\mathcal{A}}(s')$, and $s' \lhd s$;
- s is *indirectly blocked* if s is unnamed and some ancestor of s is directly blocked; and
- s is *blocked* if it is directly or indirectly blocked.

Given a normalized $\mathcal{ALCI}$ ontology $\mathcal{O}$ and an ABox $\mathcal{A}$ containing only named individuals, the hypertableau algorithm constructs a *derivation* $D = (V, E, \rho)$ for $\mathcal{O}$ and $\mathcal{A}$, where V and E are the vertices and edges of a finite tree, and ρ labels each vertex with an ABox. A vertex $v \in V$ is *closed* if $\bot \in \rho(v)$; otherwise, v is *open*. Labeling ρ must satisfy the following conditions:

- $\rho(v_0) = \mathcal{A}$ for v_0 the root of the tree;
- vertex v_0 is a leaf if v_0 is closed or no derivation rule from Table 2 is applicable to $\mathcal{A}_0 = \rho(v_0)$;

- in all other cases, the children $v_1, \dots, v_n$ of vertex v_0 are labeled by ABoxes $\mathcal{A}_1, \dots, \mathcal{A}_n$ obtained by applying one (arbitrarily chosen) derivation rule from Table 2 to $\mathcal{A}_0 = \rho(v_0)$.

The algorithm is sound and complete: for each derivation D, we have that $\mathcal{O} \cup \mathcal{A}$ is satisfiable if and only if D contains an open leaf. Thus, checking whether $\mathcal{O} \models A \sqsubseteq B$ holds can be solved using the hypertableau algorithm by constructing a derivation for $\mathcal{O} \cup \{C \sqsubseteq A,\ B \sqcap C \sqsubseteq \bot\}$ and $\{C(a)\}$ where C is a fresh atomic concept, and then checking whether each derivation leaf is closed.

Due to inverse roles, blocking is *dynamic*: an individual can change its blocking status more than once as the inference rules are applied, which is why individuals can be indirectly blocked. This, in turn, ensures that the algorithm runs in N2ExpTime: the algorithm nondeterministically constructs a tree of individuals that can have exponential depth and a linear branching factor. Hence, although the number of nonblocked and directly blocked individuals is at most exponential, the number of indirectly blocked individuals in the tree can be doubly exponential; Motik et al. [14] discuss these issues in more detail.

2.3. Fixed-Parameter Tractable Problems

We now recapitulate several definitions from *parameterized complexity* [24]. A *parameterized problem* is a set $P \subseteq \Sigma^* \times \mathbb{N}$, where Σ is a finite alphabet and $\mathbb{N}$ is the set of nonnegative integers. Each pair $\langle x, k\rangle \in \Sigma^* \times \mathbb{N}$ is called a *problem instance*; x is called the *input*; k called is the *parameter*; and $\|x\|$ is the length of x. Problem P is *fixed-parameter tractable* (FPT) if a computable function $f : \mathbb{N} \to \mathbb{N}$, a constant c, and an algorithm exist such that, given an arbitrary pair $\langle x, k\rangle \in \Sigma^* \times \mathbb{N}$, the algorithm decides $\langle x, k\rangle \in P$ in at most $f(k) \cdot \|x\|^c$ steps. FPT is the class of all fixed-parameter tractable problems.

Some authors use an alternative definition of parameterized complexity. Just like a regular problem, a parameterized problem P is defined as a subset of Σ^*. Furthermore, a *parameterization* is a function $\kappa : \Sigma^* \to \mathbb{N}$. A problem P is fixed-parameter tractable w.r.t. a parameterization κ if a computable function $f : \mathbb{N} \to \mathbb{N}$, a constant c, and an algorithm exist such that, for an arbitrary $x \in \Sigma^*$, the algorithm decides $x \in P$ in at most $f(\kappa(x)) \cdot \|x\|^c$ steps. The two definitions are clearly equivalent if, for each $x \in \Sigma^*$, value $\kappa(x)$ is computable in polynomial time. Furthermore, even if the latter is not the case, the two definitions coincide if a computable function g and a constant d exist such that, for each fixed k, one can check whether $\kappa(x) \leq k$ holds using at most $g(k) \cdot \|x\|^d$ steps; in other words, checking $\kappa(x) \leq k$ must be fixed-parameter tractable as well. The results we present in this paper involve the computation of a tree decomposition of a fixed width, which satisfies the latter condition.

2.4. Tree Decompositions and Treewidth of Hypergraphs and Propositional Problems

We recapitulate the notions of tree decompositions and treewidth [25], which were extensively used to obtain FPT results for a number of intractable graph-theoretic problems. For convenience, we extend these notions to hypergraphs by treating each hyperedge as the clique of all of its vertices, but this does not alter the definitions in any significant way.

Given a finite set N of nodes, an *N-hypergraph* $\mathcal{H}$ is a subset of 2^N; the elements of $\mathcal{H}$ are called *hyperedges*. A *tree decomposition* of $\mathcal{H}$ is a tuple $\mathcal{T} = \langle \mathcal{V}, \mathcal{E}, \mathcal{L}\rangle$ where $\mathcal{V}$ and $\mathcal{E}$ are sets of *vertices* and *edges*, respectively, of an undirected tree, and $\mathcal{L} : \mathcal{V} \to 2^N$ is a labeling of vertices with subsets of N such that the following conditions hold:

(T1) for each hyperedge $h \in \mathcal{H}$, a vertex $v \in \mathcal{V}$ exists such that $h \subseteq \mathcal{L}(v)$; and

(T2) for each element $n \in N$, the set $\{v \in \mathcal{V} \mid n \in \mathcal{L}(v)\}$ is connected in $\mathcal{E}$.

The *width* of $\mathcal{T}$ is $\mathsf{wd}(\mathcal{T}) := \mathsf{max}_{v \in \mathcal{V}} |\mathcal{L}(v)|$, and the *treewidth* of $\mathcal{H}$ is the minimum width over all tree decompositions of $\mathcal{H}$. In related literature width is actually defined as $\mathsf{wd}(\mathcal{T}) := \mathsf{max}_{v \in \mathcal{V}} |\mathcal{L}(v)| - 1$ in order to ensure that tree-like hypergraphs have width one, but we drop the term -1 from our definition for uniformity with Section 4. For a fixed integer mwd, a tree decomposition of width at most mwd can be computed in linear time [29]. This result was originally formulated for tree decompositions of graphs; however, by treating hyperedges as cliques, this result can also be straightforwardly applied to our definitions.

Lemma 1 ([29]). *Let $\mathcal{H}$ be an N-hypergraph, and let* mwd *be an integer. One can compute a tree decomposition of $\mathcal{H}$ of width at most* mwd*, or determine that such a tree decomposition does not exist, in time $O(f(\mathsf{mwd}) \cdot |N|)$ where f is a computable function. The resulting tree decomposition has at most $|\mathcal{H}|$ vertices.*

We next recapitulate how these notions can be used to obtain an FPT procedure for propositional satisfiability [26]. To unify the notation throughout this paper, we write propositional clauses as implications $K \sqsubseteq M$ where K is a conjunction of atoms, and M is a disjunction of atoms; furthermore, we consider a propositional interpretation J to be a set of atoms; finally, we write $J \models K \sqsubseteq M$ if $K \subseteq J$ implies $M \cap J \neq \emptyset$.

Let $\mathcal{N}$ be a set of propositional clauses. Then, we define $\mathcal{H}$ as the hypergraph that contains a hyperedge $K \cup M \in \mathcal{H}$ for each clause $K \sqsubseteq M \in \mathcal{N}$. Furthermore, a tree decomposition and the treewidth of $\mathcal{N}$ are defined as a tree decomposition and the treewidth, respectively, of $\mathcal{H}$. Now assume that the treewidth of $\mathcal{N}$ is bounded by some integer mwd. The following procedure checks the satisfiability of $\mathcal{N}$ and is fixed-parameter tractable w.r.t. mwd.

1. Compute a tree decomposition $\mathcal{T} = \langle \mathcal{V}, \mathcal{E}, \mathcal{L}\rangle$ of $\mathcal{H}$ of width at most mwd. By Lemma 1, this step is FPT w.r.t. mwd.

2. Associate with each vertex $v \in \mathcal{V}$ a set $\mathcal{P}_v$ of propositional interpretations as follows: set $\mathcal{P}_v := 2^{\mathcal{L}(v)}$, and then eliminate each $J \in \mathcal{P}_v$ for which a clause $K \sqsubseteq M \in \mathcal{N}$ exists such that $K \cup M \subseteq \mathcal{L}(v)$ and $J \not\models K \sqsubseteq M$. Since $|\mathcal{L}(v)| \leq \mathsf{mwd}$, this step requires time polynomial in 2^{mwd} and $|\mathcal{N}|$, and is thus FPT w.r.t. mwd.

3. Using a bottom-up approach starting from the leaves of $\mathcal{T}$, for each non-root vertex v and its parent v', eliminate from $\mathcal{P}_{v'}$ each $J' \in \mathcal{P}_{v'}$ for which no $J \in \mathcal{P}_v$ exists such that $J \cap \mathcal{L}(v) \cap \mathcal{L}(v') = J' \cap \mathcal{L}(v) \cap \mathcal{L}(v')$. We can achieve this by iterating over all pairs of $J' \in \mathcal{P}_{v'}$ and $J \in \mathcal{P}_v$; since $\mathcal{T}$ has at most $|\mathcal{N}|$ vertices by Lemma 1, this step is FPT w.r.t. mwd.

4. Set $\mathcal{N}$ is unsatisfiable if and only if $\mathcal{P}_r = \emptyset$, where r is the root of $\mathcal{T}$.

To see why this algorithm is correct, assume that $\mathcal{P}_r \neq \emptyset$; we can then construct an interpretation J for $\mathcal{N}$ as follows. Select an arbitrary propositional interpretation $J_r \in \mathcal{P}_r$; since J_r has not been deleted in step 3, for $v_1, \ldots, v_k$ the children of r, we can select interpretations $J_{v_1}, \ldots, J_{v_k}$ such that $J_r \cap \mathcal{L}(r) \cap \mathcal{L}(v_i) = J_{v_i} \cap \mathcal{L}(r) \cap \mathcal{L}(v_i)$ holds for each $1 \leq i \leq k$. By inductively repeating this argument from the root to the leaves, we associate with each vertex $v \in \mathcal{V}$ an interpretation J_v, and then we define $J = \bigcup_{v \in \mathcal{V}} J_v$. Now since $\mathcal{T}$ satisfies properties (T1) and (T2), it is straightforward to see that $J \models \mathcal{N}$. The converse direction can be shown analogously, and we omit it for the sake of brevity. Thus, the treewidth of $\mathcal{N}$ can be taken as an indicator of the "hardness" of $\mathcal{N}$, and it shows the size of the sets of atomic concepts that we need to consider to decide the satisfiability of $\mathcal{N}$.

3. Consequence-Based Framework for $\mathcal{ALCI}$

In this section we present our consequence-based framework for solving the following reasoning problem: given a normalized $\mathcal{ALCI}$ ontology $\mathcal{O}$ and a finite set of queries $\mathcal{Q}$, determine whether $\mathcal{O} \models K \sqsubseteq M$ holds for each query $K \sqsubseteq M \in \mathcal{Q}$. Note that this problem captures *ontology classification*, which involves only queries of the form $A \sqsubseteq B$ for A and B atomic concepts from $\mathcal{O}$; furthermore, by using the transformations from Section 2, the framework can also be applied to $\mathcal{SHI}$ ontologies. In Section 3.1 we discuss the intuitions behind our framework, in Section 3.2 we introduce the framework formally, in Section 3.3 we discuss certain redundancy elimination techniques that can be used to optimize reasoning, and finally in Section 3.4 we discuss several concrete instantiations of our framework.

3.1. Intuitions

In order to capture the essence of the known similar algorithms, our framework can be instantiated in many different ways by varying several parameters. In this section we describe the framework at a high level, and we discuss possible parameterizations in Sections 3.3 and 3.4. In the rest of this section, we fix the ontology $\mathcal{O}$ and the query q as specified in Example 2, and then we show how one can prove that $\mathcal{O} \models q$ holds using our framework.

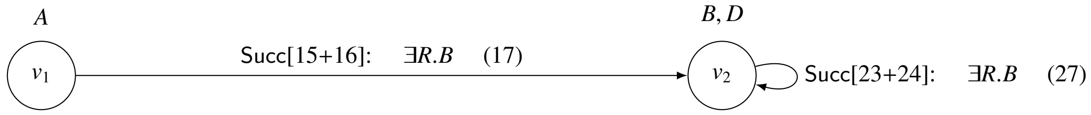
Example 2. *Let $\mathcal{O}$ be the ontology consisting of clauses* (1)–(10)*, and let $q = A \sqsubseteq G$; one can readily verify that $\mathcal{O} \models q$.*

$$A \sqsubseteq A_i \quad \text{for } 1 \leq i \leq n \quad (1) \qquad C \sqsubseteq \exists R.B \quad (6)$$
$$A_i \sqsubseteq C_i \quad \text{for } 1 \leq i \leq n \quad (2) \qquad C \sqsubseteq \forall R.D \quad (7)$$
$$B \sqsubseteq B_i \quad \text{for } 1 \leq i \leq n \quad (3) \qquad B \sqcap D \sqsubseteq E \sqcup \forall R^-.G \quad (8)$$
$$B_i \sqsubseteq C_i \quad \text{for } 1 \leq i \leq n \quad (4) \qquad E \sqsubseteq \forall R^-.F \quad (9)$$
$$C_1 \sqcap \ldots \sqcap C_n \sqsubseteq C \quad (5) \qquad A \sqcap F \sqsubseteq G \quad (10)$$

Resolution decision procedures [30] can decide whether $\mathcal{O} \models A \sqsubseteq G$ holds by transforming $\mathcal{O}$ into a set of clauses, adding clauses $A(a)$ and $\neg G(a)$ obtained from the negation of the theorem $A \sqsubseteq G$ that is to be proved, and then saturating the result using a suitable first-order resolution variant [31]. Such algorithms are typically worst-case optimal, and they can solve many practically-relevant problems; however, on complex ontologies they can easily run into a combinatorial explosion [32]. In our example, from clauses (2)–(5) resolution can derive 3^n clauses of the form $L_1 \sqcap \ldots \sqcap L_n \sqsubseteq C$ with $L_i \in \{A_i, B_i, C_i\}$, thus covering all possible combinations of atomic concepts that might be relevant. In contrast, when applied to $\mathcal{O}$, the hypertableau algorithm from Section 2.2 does not run into this problem: the algorithm is initialized using the fact $A(a)$, and then it derives $A_i(a)$ and $C_i(a)$ for each $1 \leq i \leq n$ and $C(a)$; thus, the algorithm does not consider the "irrelevant" combinations of A_i, B_i, and C_i.

The hypertableau algorithm can thus be seen as being more "goal directed" than resolution. However, as we discussed in Section 2.2, the hypertableau algorithm can construct a very large tree of individuals most of which are indirectly blocked, and can thus perform a lot of redundant computation since each indirectly-blocked individual is a "copy" of a nonblocked or a directly blocked individual. Resolution is not susceptible to such problems: clauses introduced by resolution are universally quantified and are (unlike individuals in the tableau algorithm) not localized to a specific part of a model. Resolution can thus describe all relevant combinations of atomic concepts using exponentially many clauses, thereby avoiding redundant computation.

Our consequence-based algorithm can be understood as a hybrid between resolution and the hypertableau algorithm. It does not explicitly construct a model; instead, it uses resolution to compute clauses that describe a model $\mathcal{I}$ of $\mathcal{O}$ refuting the relevant


A
v_1
Succ[15+16]: $\exists R.B$ (17)
B, D
v_2
Succ[23+24]: $\exists R.B$ (27)

Initialization:	$\top \sqsubseteq A$		(11)
Hyper[1+11]:	$\top \sqsubseteq A_i$	for $1 \leq i \leq n$	(12)
Hyper[2+12]:	$\top \sqsubseteq C_i$	for $1 \leq i \leq n$	(13)
Hyper[5+13]:	$\top \sqsubseteq C$		(14)
Hyper[6+14]:	$\top \sqsubseteq \exists R.B$		(15)
Hyper[7+14]:	$\top \sqsubseteq \forall R.D$		(16)
Pred[15+17+26]:	$\top \sqsubseteq F \sqcup G$		(28)
Hyper[10+11+28]:	$\top \sqsubseteq G$		(30)

Succ[15+16]:	$\top \sqsubseteq B$		(18)
Succ[15+16]:	$\top \sqsubseteq D$		(19)
Hyper[3+18]:	$\top \sqsubseteq B_i$	for $1 \leq i \leq n$	(20)
Hyper[4+20]:	$\top \sqsubseteq C_i$	for $1 \leq i \leq n$	(21)
Hyper[5+21]:	$\top \sqsubseteq C$		(22)
Hyper[6+22]:	$\top \sqsubseteq \exists R.B$		(23)
Hyper[7+22]:	$\top \sqsubseteq \forall R.D$		(24)
Hyper[8+18+19]:	$\top \sqsubseteq E \sqcup \forall R^{-}.G$		(25)
Hyper[9+25]:	$\top \sqsubseteq \forall R^{-}.F \sqcup \forall R^{-}.G$		(26)
Pred[23+27+26]:	$\top \sqsubseteq F \sqcup G$		(29)

Figure 1: Example Inferences of the Consequence-Based Algorithm

queries in $\mathcal{Q}$. The inferences of the algorithm are not localized to a part of $\mathcal{I}$, which allows our algorithm to avoid redundant computation: although there are many technical differences, our algorithm is somewhat related to the tableau algorithms with caching [33, 34]. To be goal oriented, our algorithm constructs a *context structure*—a graph-like structure whose vertices are called *contexts*. Intuitively, each context describes one or more elements in the model $\mathcal{I}$, and the edges between contexts capture the relations between the corresponding elements of $\mathcal{I}$. Each context v is associated with a set $\mathsf{core}(v)$ of *core* atomic concepts. Intuitively, the concepts in $\mathsf{core}(v)$ hold for each element of $\mathcal{I}$ that corresponds to v; hence, $\mathsf{core}(v)$ determines the "type" of v and the corresponding model elements. Furthermore, each context v is associated with a set of clauses $\mathcal{S}(v)$ that, as in propositional resolution, describe the domain elements in $\mathcal{I}$ that correspond to v. Each clause $K \sqsubseteq M \in \mathcal{S}(v)$ is "relative" to $\mathsf{core}(v)$ and should be interpreted as $\mathsf{core}(v) \sqcap K \sqsubseteq M$. In other words, the concepts in $\mathsf{core}(v)$ hold in all model elements corresponding to v, so we drop $\mathsf{core}(v)$ from the clauses in $\mathcal{S}(v)$ for the sake of clarity. We next demonstrate how to use the derivation rules in Table 3 on page 13 to prove $\mathcal{O} \models A \sqsubseteq G$. We will construct the context structure shown in Figure 1; each context is shown as a circle, the core of each context is shown above the circle, the clauses belonging to the context are shown below the circle, and the numbers next to the clauses correspond to the order of inference rule applications.

To avoid the drawbacks outlined earlier, our algorithm does not perform inferences between the clauses in $\mathcal{O}$; instead, each inference involves either a single set of clauses $\mathcal{S}(v)$ and possibly the ontology $\mathcal{O}$, or a pair of sets of clauses $\mathcal{S}(v)$ and $\mathcal{S}(u)$. To initiate the inference process, we initialize the algorithm according to the target query. Since our goal is to prove $\mathcal{O} \models A \sqsubseteq G$, we introduce a single context v_1 with $\mathsf{core}(v_1) = \{A\}$, and we add clause (11) to $\mathcal{S}(v_1)$. Intuitively, this says that the model $\mathcal{I}$ must contain at least one element in which A holds, which is similar to initializing the hypertableau algorithm by $A(a)$. We then use the Hyper rule to derive clauses (12)–(16). Since only A is assumed to hold in context v_1, we derive only a linear number of clauses: no B_i holds in v_1, so we do not derive clauses with irrelevant combinations of atomic concepts. These inferences are analogous to the inferences of the Hyp-rule in the hypertableau algorithm, with the difference that the conclusions are not localized: they hold for each element of $\mathcal{I}$ that corresponds to v_1.

Clause (15) says that the elements in $\mathcal{I}$ corresponding to v_1 must have a successor in which B holds. To satisfy this requirement, we use the Succ rule to introduce context v_2 and add the edge (17) from v_1 to v_2; the edge is labeled with the concept $\exists R.B$ that it satisfies. The Succ rule combines the $\exists$-rule and the $\forall^{+}$-rule from the hypertableau algorithm by means of sets of atomic concepts $\mathbf{B}_k$ and $\mathbf{B}_p$. Set $\mathbf{B}_k$ contains atomic concepts known to hold in v_2 due to universal restrictions—that is, those concepts L for which $\top \sqsubseteq \forall R.L$ has been derived in v_1. In our example, clause (16) ensures that D holds in v_2, and so we have $\mathbf{B}_k = \{D\}$. Thus, set $\{B\} \cup \mathbf{B}_k = \{B, D\}$ provides us with an "upper bound" on the core of the new context: both B and D necessarily hold in context v_2, but we can use an arbitrary subset of $\{B, D\}$ as the core of v_2. Choosing smaller cores might reduce the number of contexts because it allows us to reuse the same context to satisfy different applications of the Succ rule, but it might also increase the number of clauses per context as the latter is then "less precise." The decision which subset to use is determined by a *strategy*, which is supplied as parameter to our algorithm. In this example we use the *eager* strategy that always uses the maximal subset and thus sets $\mathsf{core}(v_2) = \{B, D\}$; however, we discuss other reasonable strategies and the tradeoffs involved in the choice of a core in more detail in Section 3.4. The Succ rule also initializes $\mathcal{S}(v_2)$ with clauses (18) and (19) specifying that B and D hold in each element of $\mathcal{I}$ that is represented by context v_2. Set $\mathbf{B}_p$ contains atomic concepts that can possibly hold in v_2 due to universal restrictions—that is, those concepts L for which $K \sqsubseteq M \sqcup \forall R.L$ has been derived in v_1 for some K and M. Thus, set $\{B\} \cup \mathbf{B}_p$ provides us with an "upper bound" on the concepts that might need to be considered in v_2. Since (16) is the only clause in $\mathcal{S}(v_1)$ containing a universal restriction, in our example we have $\mathbf{B}_p = \mathbf{B}_k = \{D\}$. Sets $\mathbf{B}_k$ and $\mathbf{B}_p$ need not coincide in general, and the Succ rule adds a clause of the form $L \sqsubseteq L$ for each concept L from $\mathbf{B}_p$ that is not in the core of the new context.

We next use the Hyper rule to derive clauses (20)–(26) in $\mathcal{S}(v_2)$. As in the hypertableau algorithm, since only B holds in context v_2, we do not derive a clause involving A_i. Now clause (23) requires an edge labeled with $\exists R.B$; due to clause (24), the core of the target context can be any subset of $\{B, D\}$; and due to the eager strategy, the core of the target context will be $\{B, D\}$. But then, there is no need to introduce a fresh context: our eager strategy can instruct the Succ rule to reuse the existing context v_2 since we already have $\mathsf{core}(v_2) = \{B, D\}$. Consequently, we introduce edge (27) using the Succ rule. This "reuse" of v_2 is similar to blocking in the hypertableau algorithm, but is actually much more effective in eliminating redundant computations. First, we never need more than exponentially many contexts, whereas the hypertableau algorithm can constructs trees of doubly exponential size. Second, the clauses belonging to each context are not localized to a specific place in $\mathcal{I}$, and so our algorithm draws the inferences for a particular core only once. In contrast, to ensure that the labels of s and s' coincide and thus ensure the blocking condition, the hypertableau algorithm must draw the same conclusions for s and s'; furthermore, an individual can directly block exponentially many other individuals, so the potential for redundant work is even higher.

Clause (26) says that each element in $\mathcal{I}$ corresponding to a predecessor of an element represented by v_2 must satisfy F or G; hence, we apply the Pred rule to edge (17) and clauses (15) and (26) to derive clause (28). Furthermore, we also apply the Pred rule to edge (27) and thus derive clause (29). The Pred rule essentially "pulls" the information from the successor to the predecessor; however, unlike the $\forall^-$-rule in the hypertableau algorithm, it simultaneously deals with several universal restrictions.

Finally, we use the Hyper rule to derive clause (30), at which point no further inferences are possible. Since all clauses are "relative" to the core of the corresponding context, clause (30) actually corresponds to $A \sqsubseteq G$, so we have proved $\mathcal{O} \models A \sqsubseteq G$. In fact, due to (30), we know that $\mathcal{O} \models K \sqsubseteq M$ for each query $K \sqsubseteq M$ such that $A \sqsubseteq G$ is a strengthening of $K \sqsubseteq M$. Our algorithm is thus not just refutationally complete: for each query $K \sqsubseteq M$ such that $\mathcal{O} \models K \sqsubseteq M$, it derives at least one strengthening of $K \sqsubseteq M$ in each context that *covers* (cf. Definition 4) the query.

We finish this section with a note that unrestricted application of the Hyper rule can be very prolific, so we use the *ordered* resolution variant [31]: we parameterize our algorithm with an ordering on literals, and, for each clause $K \sqsubseteq M \in \mathcal{S}(v)$, we apply the inference rules only to literals that are maximal in M w.r.t. the ordering. The Pred rule, however, introduces a complication: clause (26), which is needed for the Pred rule, can be derived from (25) only if $\forall R^-.G$ is smaller than E in the ordering. Therefore, we use an ordering $\prec_v$ per context v; furthermore, if v has an incoming edge labeled with $\exists R.A$, then $\prec_v$ must be *R-admissible* (cf. Definition 3)—that is, each literal of the form $\forall \mathsf{inv}(R).B$ must be smallest in $\prec_v$. These restrictions ensure that all clauses that can participate in the Pred rule, such as (26), are derived in $\mathcal{S}(v)$ and so the rule can be applied. Please note that orderings are optional: the trivial empty ordering (i.e., the ordering under which all literals are incomparable) is R-admissible for each role R, and it can therefore be used in each context. Moreover, orderings are not necessary for the fixed-parameter tractability results we present in Sections 4 and 5: these hold even if each context uses the trivial empty ordering.

3.2. Formalizing the Consequence-Based Framework

In this section we formalize the intuitions that we discussed in Section 3.1. Recall that in Section 2.1 we introduced Σ_L as the set of all literals, and $\Sigma_L^\exists$ as the set of all existential restrictions over the signature $\Sigma = \langle \Sigma_A, \Sigma_T \rangle$. Moreover, recall that we often treat conjunctions and disjunctions of literals as sets of their respective conjuncts and disjuncts. We first introduce the notion of a literal ordering, which we will use later to restrict applicability of our inference rules.

Definition 3. *A* literal ordering $\prec$ *is a strict partial order (i.e., an irreflexive and transitive relation) on the set* Σ_L *of all literals. A literal* $L \in \Sigma_L$ *is* $\prec$-minimal *if no literal* $L' \in \Sigma_L$ *exists such that* $L' \prec L$*; moreover, a set of literals* N *is* $\prec$-minimal *if each literal in* N *is $\prec$-minimal. A literal* $L \in \Sigma_L$ *is* $\prec$-maximal *w.r.t. a set of literals* N*, written* $L \not\prec N$*, if no literal* $L' \in N$ *exists such that* $L \prec L'$*. Given a role* R*, ordering* $\prec$ *is* R-admissible *if each literal of the form* $\forall \mathsf{inv}(R).A$ *is $\prec$-minimal.*

By Definition 3, a literal ordering $\prec$ is a partial order so two literals can be incomparable w.r.t. $\prec$. Thus, a literal L being maximal in some set of literals N does not imply that $L' \prec L$ holds for each literal $L' \in N$; instead, this only means that no literal in N is larger than L w.r.t. $\prec$, which we reflect by using notation $L \not\prec N$ for the notion of maximality.

Throughout the rest of this paper we fix a "global" countably infinite set of *contexts* $\mathbf{X}$. Sets $\mathbf{X}$, Σ_L, and $\Sigma_L^\exists$ provide us with building blocks for the following definition of a context structure.

Definition 4. *A* context structure *is a tuple* $\mathcal{D} = \langle \mathcal{V}, \mathcal{E}, \mathsf{core}, \prec \rangle$*, where* $\mathcal{V} \subseteq \mathbf{X}$ *is a finite set of contexts,* $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V} \times \Sigma_L^\exists$ *is a finite set of edges labeled by existential restrictions, function* $\mathsf{core} : \mathcal{V} \to 2^{\Sigma_L}$ *labels each context with a finite set of literals, and function* $\prec$ *assigns to each context* $v \in \mathcal{V}$ *a literal ordering* $\prec_v$*. Such* $\mathcal{D}$ *is* admissible *if, for each edge* $\langle v, u, \exists R.A \rangle \in \mathcal{E}$*, the literal ordering* $\prec_u$ *is R-admissible. Furthermore,* $\mathcal{D}$ *is* over *a set of literals* $\mathbf{L}$ *if all literals in* $\mathcal{D}$ *are contained in* $\mathbf{L}$*. Finally, given a context* v*, set* $\mathsf{core}(v)$ *is often treated as the conjunction of its atomic concepts, thus allowing* $\mathsf{core}(v)$ *to occur in concepts and axioms.*

Let $v \in \mathcal{V}$ *be an arbitrary context of* $\mathcal{D}$*; then,* v *is* trivial *if* $\mathsf{core}(v) = \emptyset$ *and* $\prec_v = \emptyset$*. In addition, let* $K \sqsubseteq M$ *be an arbitrary query; then,* v *is* sound *for* $K \sqsubseteq M$ *if* $\mathsf{core}(v) \subseteq K$*;* v *is* complete *for* $K \sqsubseteq M$ *if* M *is* $\prec_v$*-minimal; and* v covers $K \sqsubseteq M$ *if* v *is both sound and complete for* $K \sqsubseteq M$*.*

A clause system, which we formalize next, assigns a set of clauses $\mathcal{S}(v)$ to each context v. As we discussed in Section 3.1, our consequence-based algorithm produces a context structure $\mathcal{D}$ and a clause system $\mathcal{S}$. The clauses in each set $\mathcal{S}(v)$ will be "relative" to $\mathsf{core}(v)$—that is, for each clause $K' \sqsubseteq M' \in \mathcal{S}(v)$, we will have $\mathcal{O} \models \mathsf{core}(v) \sqcap K' \sqsubseteq M'$. Moreover, we will be able to use $\mathcal{S}$ to decide query entailment: for each query $K \sqsubseteq M$ and each context v covering $K \sqsubseteq M$ initialized so that $K \sqsubseteq L \hat{\in} \mathcal{S}(v)$ holds for each literal $L \in K$, we will have $\mathcal{O} \models K \sqsubseteq M$ if and only if $K \sqsubseteq M \hat{\in} \mathcal{S}(v)$. The notion of context soundness in Definition 4 ensures that query $K \sqsubseteq M$ is compatible with the intended reading of the clauses in $\mathcal{S}(v)$, and the notion of context completeness ensures that ordering $\prec_v$ does not prevent the derivation of a strengthening of $K \sqsubseteq M$ in such $\mathcal{S}(v)$.

Definition 5. *A* clause system *for a context structure $\mathcal{D} = \langle \mathcal{V}, \mathcal{E}, \mathsf{core}, \prec \rangle$ is a function $\mathcal{S}$ that assigns to each context $v \in \mathcal{V}$ a finite set of clauses $\mathcal{S}(v)$.*

To simplify the presentation, in the rest of this section we fix a normalized $\mathcal{ALCI}$ ontology $\mathcal{O}$ and a finite set of queries $\mathcal{Q}$—that is, we assume that all subsequent definitions and theorems in this section are implicitly parameterized with $\mathcal{O}$ and $\mathcal{Q}$. Furthermore, we fix $\mathbf{L}$ to be the set of literals occurring in $\mathcal{O} \cup \mathcal{Q}$; since $\mathcal{O}$ and $\mathcal{Q}$ are finite sets, set $\mathbf{L}$ is finite as well.

As we explained in Section 3.1, our algorithm is based on the inference rules shown in Table 3. The Hyper, Succ, and Pred rules are responsible for completeness, and we discussed the intuitions behind these rules in Section 3.1. The Elim rule supports redundancy elimination and is not needed for completeness or the complexity results in Sections 4 and 5. We discuss the intuitions underlying the Elim rule in more detail in Section 3.3; however, in order to present the soundness, completeness, and termination proofs in one place, we introduce the rule here. The completeness of our algorithm is guaranteed by Theorem 6, the proof of which is given in Appendix A. The theorem essentially says that, given a context structure $\mathcal{D}$ and a clause system $\mathcal{S}$ saturated under the the Hyper, Pred, and Succ rules, we can read off the consequences of the form $K \sqsubseteq M$ from the sets $\mathcal{S}(v)$ provided that $K \sqsubseteq M$ is a query, the context v is complete (cf. Definition 4) for $K \sqsubseteq M$, and set $\mathcal{S}(v)$ is appropriately initialized. The initialization is similar to asserting $L(a)$ for each literal $L \in K$ at the beginning of a hypertableau test of $\mathcal{O} \models K \sqsubseteq M$. Please note that Theorem 6 depends on the Succ rule being *not applicable*, and so the exact definition of strategy used in the postcondition of the Succ rule is not relevant for completeness. In the rest of this section we discuss how to initialize the sets $\mathcal{S}(v)$ and how to satisfy the precondition of the Succ rule to obtain a sound and complete algorithm.

Theorem 6 (Completeness). *Let $\mathcal{D} = \langle \mathcal{V}, \mathcal{E}, \mathsf{core}, \prec \rangle$ be an admissible context structure, and let $\mathcal{S}$ be a clause system for $\mathcal{D}$ such that the* Hyper, Pred, *and* Succ *rules from Table 3 are not applicable to $\mathcal{D}$ and $\mathcal{S}$. Then, $K \sqsubseteq M \hat{\in} \mathcal{S}(v)$ holds for each query $K \sqsubseteq M$ and each context $v \in \mathcal{V}$ that satisfy all of the following three conditions:*

- *$\mathcal{O} \models K \sqsubseteq M$,*
- *context v is complete for query $K \sqsubseteq M$, and*
- *$K \sqsubseteq L \hat{\in} \mathcal{S}(v)$ for each literal $L \in K$.*

As we discussed in Section 3.1, our framework is parameterized by a function strategy, which we formalize in Definition 7. Intuitively, when the Succ rule identifies a context v and an existential restriction $\exists R.A$ for which the rule's precondition is not satisfied, $\mathsf{strategy}(\exists R.A, \mathbf{B}_k, \mathcal{D})$ returns a triple $\langle u, \mathsf{core}', \prec' \rangle$ that specifies how to modify $\mathcal{D}$ and $\mathcal{S}$ so that the precondition becomes satisfied. The strategy has two options. First, the strategy can decide to extend $\mathcal{D}$ by returning a fresh context u; in such a case, u is added to $\mathcal{D}$, and core' and $\prec'$ specify how to initialize $\mathsf{core}(u)$ and $\prec_u$. Second, the strategy can decide to reuse a context u that is already a part of $\mathcal{D}$; if so, then ordering $\prec_u$ is intersected with $\prec'$ to ensure R-admissibility, but also preserve admissibility for all other roles. Definition 7 requires each strategy to be bounded: if the signature is finite, then the strategy should return only finitely many different results on all possible inputs. Please note that a finite signature does not necessarily bound the size of $\mathcal{D}$, so the number of different arguments (and return values) for strategy is not bounded. Bounded strategies can thus introduce only finitely many fresh contexts, which ensures termination.

Definition 7. *An* expansion strategy *is a function* strategy *computable in polynomial time that takes an existential restriction $\exists R.A$, a set of atomic concepts $\mathbf{B}_k$, and a context structure $\mathcal{D} = \langle \mathcal{V}, \mathcal{E}, \mathsf{core}, \prec \rangle$. The result of* $\mathsf{strategy}(\exists R.A, \mathbf{B}_k, \mathcal{D})$ *is a triple $\langle u, \mathsf{core}', \prec' \rangle$ where*

- core' *is a subset of $\{A\} \cup \mathbf{B}_k$,*
- *$\prec'$ is an R-admissible literal ordering, and*
- *either $u \in \mathbf{X} \setminus \mathcal{V}$ is a fresh context, or $u \in \mathcal{V}$ is a context in $\mathcal{D}$ such that* $\mathsf{core}(u) = \mathsf{core}'$.

Each expansion strategy must be bounded, *which is the case if, for each finite set of literals $\mathbf{L}$, the number* $|\mathsf{strategy}|_{\mathbf{L}}$ *of different values that* strategy *can return for all possible arguments over $\mathbf{L}$ is finite.*

We next define when a context structure $\mathcal{D}$ and a clause system $\mathcal{S}$ are sound for $\mathcal{O}$. Recall that, for a context v, clause $K \sqsubseteq M$ in $\mathcal{S}(v)$ is "relative" to $\mathsf{core}(v)$—that is, the clause should be interpreted as $\mathsf{core}(v) \sqcap K \sqsubseteq M$. Soundness for a clause simply means that $\mathcal{O} \models \mathsf{core}(v) \sqcap K \sqsubseteq M$ should hold for each clause $K \sqsubseteq M \in \mathcal{S}(v)$. To understand the notion of soundness for a context structure, consider an arbitrary edge $\langle v, u, \exists R.A \rangle$ in $\mathcal{D}$. As we explained in Section 3.1, the Pred rule "pulls" information from $\mathcal{S}(u)$ into $\mathcal{S}(v)$; however, the clauses in $\mathcal{S}(v)$ and $\mathcal{S}(u)$ are "relative" to $\mathsf{core}(v)$ and $\mathsf{core}(u)$, respectively, so $\mathsf{core}(u)$ and

$\mathsf{core}(v)$ must be related as specified in Definition 8 to make the Pred rule sound. Please remember that $\mathsf{core}(u)$ and $\mathsf{core}(v)$ should be understood as conjunctions of the respective sets of atomic concepts.

Definition 8. *A context structure $\mathcal{D} = \langle \mathcal{V}, \mathcal{E}, \mathsf{core}, \prec \rangle$ is* sound *for $\mathcal{O}$ if $\mathcal{O} \models \mathsf{core}(v) \sqcap \exists R.A \sqsubseteq \exists R.[\mathsf{core}(u) \sqcap A]$ holds for each edge $\langle v, u, \exists R.A \rangle \in \mathcal{E}$. Furthermore, a clause system $\mathcal{S}$ is* sound *for $\mathcal{O}$ if $\mathcal{O} \models \mathsf{core}(v) \sqcap K \sqsubseteq M$ holds for each context $v \in \mathcal{V}$ and each clause $K \sqsubseteq M \in \mathcal{S}(v)$.*

Proposition 9 shows that the inference rules in Table 3 preserve admissibility and soundness of a context structure and a clause system. The former follows immediately from Definition 7, and the proof of the latter is analogous to the soundness proof for first-order resolution [31]: we show that each clause introduced by an inference rule is a logical consequence of $\mathcal{O}$.

Proposition 9 (Soundness)**.** *Let $\mathcal{D}_1 = \langle \mathcal{V}, \mathcal{E}, \mathsf{core}, \prec \rangle$ be a context structure, let $\mathcal{S}_1$ be a clause system for $\mathcal{D}_1$, and let $\mathcal{D}_2$ and $\mathcal{S}_2$ be the context structure and the clause system, respectively, obtained by applying an inference rule from Table 3 to $\mathcal{D}_1$ and $\mathcal{S}_1$. If $\mathcal{D}_1$ is admissible, then $\mathcal{D}_2$ is admissible. Furthermore, if both $\mathcal{D}_1$ and $\mathcal{S}_1$ are sound for $\mathcal{O}$, then both $\mathcal{D}_2$ and $\mathcal{S}_2$ are sound for $\mathcal{O}$.*

Proof. Assume that $\mathcal{D}_1$ is admissible and that the Succ rule is applied to $\mathcal{D}_1$ as shown in Table 3. By Definition 7, literal ordering $\prec'$ is R-admissible; furthermore, for arbitrary literal orderings $\prec_1$ and $\prec_2$ that are R- and S-admissible, respectively, set $\prec_1 \cap \prec_2$ is a literal ordering that is both R- and S-admissible. Therefore, regardless of whether context u is fresh or not, $\mathcal{D}_2$ is admissible. Now assume that both $\mathcal{D}_1$ and $\mathcal{S}_1$ are sound; we next show that both $\mathcal{D}_2$ and $\mathcal{S}_2$ are sound as well. To this end, let $\mathcal{I} = \langle \Delta^{\mathcal{I}}, \cdot^{\mathcal{I}} \rangle$ be an arbitrary model of $\mathcal{O}$, and consider all possible inference rules that derive a clause in $\mathcal{S}_2$ or modify $\mathcal{D}_2$.

(Hyper rule) Assume that the rule is applied as in Table 3; we will show that $\mathcal{O} \models \mathsf{core}(v) \sqcap \bigsqcap_{i=1}^{n} K_i \sqsubseteq M \sqcup \bigsqcup_{i=1}^{n} M_i$. To this end, consider an arbitrary element $\delta \in \Delta^{\mathcal{I}}$ such that $\delta \in (\mathsf{core}(v) \sqcap \bigsqcap_{i=1}^{n} K_i)^{\mathcal{I}}$. Now $\mathcal{S}_1$ is sound for $\mathcal{O}$ so, for each $1 \leq i \leq n$, we have $\mathcal{O} \models \mathsf{core}(v) \sqcap K_i \sqsubseteq M_i \sqcup A_i$, which implies $\delta \in (M_i \sqcup A_i)^{\mathcal{I}}$. If $\delta \in M_i^{\mathcal{I}}$ for some $1 \leq i \leq n$, then clearly $\delta \in (\bigsqcup_{i=1}^{n} M_i)^{\mathcal{I}}$. Otherwise, we have $\delta \in (\bigsqcap_{i=1}^{n} A_i)^{\mathcal{I}}$, but then $\bigsqcap_{i=1}^{n} A_i \sqsubseteq M \in \mathcal{O}$ implies $\delta \in M^{\mathcal{I}}$. In either case, we have $\delta \in (M \sqcup \bigsqcup_{i=1}^{n} M_i)^{\mathcal{I}}$. Since δ was chosen arbitrarily, $\mathcal{I}$ satisfies the conclusion of the rule, as required.

(Pred rule) Assume that the rule is applied as in Table 3; we will show that $\mathcal{O} \models \mathsf{core}(v) \sqcap \bigsqcap_{i=0}^{n} K_i \sqsubseteq \bigsqcup_{i=0}^{n} M_i \sqcup \bigsqcup_{j=1}^{m} C_j$. To this end, consider an arbitrary element $\delta \in \Delta^{\mathcal{I}}$ such that $\delta \in (\mathsf{core}(v) \sqcap \bigsqcap_{i=0}^{n} K_i)^{\mathcal{I}}$. As in the previous case, if $\delta \in M_i^{\mathcal{I}}$ for some $0 \leq i \leq n$, then clearly $\delta \in (\bigsqcup_{i=0}^{n} M_i)^{\mathcal{I}}$. Otherwise, as $\mathcal{S}_1$ is sound for $\mathcal{O}$, we have $\delta \in (\exists R.A)^{\mathcal{I}}$ and $\delta \in (\forall R.B_i)^{\mathcal{I}}$ for each $1 \leq i \leq n$. Since $\langle v, u, \exists R.A \rangle \in \mathcal{E}$ and $\mathcal{D}_1$ is sound, we have $\mathcal{O} \models \mathsf{core}(v) \sqcap \exists R.A \sqsubseteq \exists R.[\mathsf{core}(u) \sqcap A]$; therefore, since $\delta \in [\mathsf{core}(v) \sqcap \exists R.A]^{\mathcal{I}}$, we have $\delta \in (\exists R.[\mathsf{core}(u) \sqcap A])^{\mathcal{I}}$. Hence, an element $\gamma \in \Delta^{\mathcal{I}}$ exists such that $\langle \delta, \gamma \rangle \in R^{\mathcal{I}}$ and $\gamma \in [\mathsf{core}(u) \sqcap A \sqcap \bigsqcap_{i=1}^{n} B_i]^{\mathcal{I}}$. But then, due to either of the two alternative premises of the rule, we also have $\gamma \in [\bigsqcup_{j=1}^{m} \forall \mathsf{inv}(R).C_j]^{\mathcal{I}}$; thus, $\delta \in (\bigsqcup_{j=1}^{m} C_j)^{\mathcal{I}}$ holds. Since δ was chosen arbitrarily, $\mathcal{I}$ satisfies the conclusion of the rule, as required.

(Succ rule) Assume that the rule is applied as in Table 3; in the rest of this proof, we consider set $\mathbf{B}_k$ to be equivalent to the conjunction of the atomic concepts contained in it. Each clause introduced by the rule is of the form $L \sqsubseteq L$, so clearly $\mathcal{I} \models \mathsf{core}(u) \sqcap L \sqsubseteq L$. We next show that the new edge $\langle v, u, \exists R.A \rangle$ introduced by the rule satisfies the axiom from Definition 8. For each atomic concept $B \in \mathbf{B}_k$, we have $\top \sqsubseteq \forall R.B \in \mathcal{S}(v)$ by the rule preconditions, so $\mathcal{O} \models \mathsf{core}(v) \sqsubseteq \forall R.B$ since $\mathcal{S}_1$ is sound for $\mathcal{O}$; but then, we have $\mathcal{O} \models \mathsf{core}(v) \sqcap \exists R.A \sqsubseteq \exists R.[A \sqcap \mathbf{B}_k]$. Furthermore, since $\mathsf{core}(u) = \mathsf{core}' \subseteq A \sqcap \mathbf{B}_k$ holds, we also have $\mathcal{O} \models \mathsf{core}(v) \sqcap \exists R.A \sqsubseteq \exists R.[\mathsf{core}(u) \sqcap A]$, as required. □

The following notion of a covering mapping maps each query $q \in \mathcal{Q}$ to a context in a context structure. Intuitively, our algorithm will use such a mapping to find for each query $q \in \mathcal{Q}$ a context that can be used to verify $\mathcal{O} \models q$.

Definition 10. *Let $\mathcal{D} = \langle \mathcal{V}, \mathcal{E}, \mathsf{core}, \prec \rangle$ be a context structure. A* covering mapping *from $\mathcal{Q}$ to $\mathcal{D}$ is a function $\vartheta : \mathcal{Q} \to \mathcal{V}$ such that each query $q \in \mathcal{Q}$ is covered in the context $\vartheta(q)$.*

We are finally ready to formally define our consequence-based reasoning algorithm. Apart from $\mathcal{O}$ and $\mathcal{Q}$, the algorithm is parameterized by an expansion strategy strategy, a context structure $\mathcal{D}$ with no edges, and a covering mapping ϑ. By passing $\mathcal{D}$ and ϑ as arguments, the algorithm's users can decide how to initialize the contexts necessary for checking the queries in $\mathcal{Q}$; we discuss reasonable possibilities in Section 3.4.1. Since the algorithm is given a context structure without edges, $\mathcal{D}$ is trivially sound for $\mathcal{O}$; by inductively applying Proposition 9, the resulting clause system is sound for $\mathcal{O}$. Furthermore, the algorithm is complete because steps 1 and 2 satisfy the preconditions of Theorem 6.

Algorithm 11. *The* consequence-based reasoning algorithm *for $\mathcal{ALCI}$ takes $\mathcal{O}$, $\mathcal{Q}$, a context structure $\mathcal{D} = \langle \mathcal{V}, \mathcal{E}, \mathsf{core}, \prec \rangle$ over $\mathbf{L}$ with $\mathcal{E} = \emptyset$, an expansion strategy* strategy*, and a covering mapping ϑ from $\mathcal{Q}$ to $\mathcal{D}$. The algorithm (nondeterministically) extends $\mathcal{D}$ and computes a clause system $\mathcal{S}$ for the extended context structure as follows.*

1. *Set $\mathcal{S}(v) := \{\top \sqsubseteq L \mid L \in \mathsf{core}(v)\}$ for each context $v \in \mathcal{V}$.*

2. *For each query $K \sqsubseteq M \in \mathcal{Q}$, let $v := \vartheta(K \sqsubseteq M)$; then, for each literal $L \in K \setminus \mathsf{core}(v)$, add $L \sqsubseteq L$ to $\mathcal{S}(v)$.*

3. *Exhaustively apply the inference rules from Table 3.*

Please note that Algorithm 11 is nondeterministic: the conclusion of an inference rule is added to $\mathcal{S}(v)$ only if $\mathcal{S}(v)$ does not contain a strengthening of the conclusion, and the order of inference rule application is not predetermined; therefore, the resulting $\mathcal{D}$ and $\mathcal{S}$ are not unique. This, however, is don't-care nondeterminism: any order of inference rule applications suffices for soundness and completeness.

In Proposition 13 we formalize the soundness and completeness argument outlined earlier. Towards this goal, in Lemma 12 we show that the Elim rule never deletes the "relevant" consequences derived by our algorithm: whenever a clause $K \sqsubseteq M$ is deleted from some set $\mathcal{S}(v)$, after deletion the set still contains a strengthening of the deleted clause.

Lemma 12. *If, at some point in the execution of Algorithm 11, $K \sqsubseteq M \mathrel{\hat{\in}} \mathcal{S}(v)$ holds for some clause $K \sqsubseteq M$ and context v, then $K \sqsubseteq M \mathrel{\hat{\in}} \mathcal{S}(v)$ holds at all future points as well.*

Proof. The proof is by a straightforward induction on the application of the rules. In particular, rules Hyper, Pred, and Succ just add clauses, so their application clearly preserves this property. Furthermore, the Elim rule removes a clause $K_2 \sqsubseteq M_2$ from $\mathcal{S}(v)$ only if $K_1 \sqsubseteq M_1 \in \mathcal{S}(v)$ where $K_1 \sqsubseteq M_1$ is a strengthening of $K_2 \sqsubseteq M_2$; since the strengthening relation on clauses is transitive, an application of the Elim rule clearly preserves this property as well. □

Proposition 13. *Let $\mathcal{S}$ be a clause system obtained by applying Algorithm 11 to $\mathcal{O}$, $\mathcal{Q}$, a context structure $\mathcal{D}$, an expansion strategy* strategy*, and a covering mapping ϑ. Then, for each query $q \in \mathcal{Q}$, we have $\mathcal{O} \models q$ if and only if $q \mathrel{\hat{\in}} \mathcal{S}(\vartheta(q))$.*

Proof. Consider an arbitrary query $K \sqsubseteq M \in \mathcal{Q}$ and let $v := \vartheta(K \sqsubseteq M)$. Context v covers $K \sqsubseteq M$, so v is sound for $K \sqsubseteq M$ and thus $\mathsf{core}(v) \subseteq K$ holds by Definition 4, and v is also complete for $K \sqsubseteq M$.

Assume that $K \sqsubseteq M \mathrel{\hat{\in}} \mathcal{S}(v)$, so a clause $K' \sqsubseteq M' \in \mathcal{S}(v)$ exists such that $K' \subseteq K$ and $M' \subseteq M$. For each clause $\top \sqsubseteq L$ added to some $\mathcal{S}(u)$ in step 1, we have $\mathcal{O} \models \mathsf{core}(u) \sqsubseteq L$ since $L \in \mathsf{core}(u)$; furthermore, for each clause $L \sqsubseteq L$ added to $\mathcal{S}(u)$ in step 2, we clearly have $\mathcal{O} \models \mathsf{core}(u) \sqcap L \sqsubseteq L$; finally, Algorithm 11 is applied to a sound context structure, so $\mathcal{D}$ and $\mathcal{S}$ are sound for $\mathcal{O}$ by Proposition 9, and $\mathcal{O} \models \mathsf{core}(v) \sqcap K' \sqsubseteq M'$ holds. Finally, $K' \cup \mathsf{core}(v) \subseteq K$ and $M' \subseteq M$ imply $\mathcal{O} \models K \sqsubseteq M$, as required.

Conversely, assume that $\mathcal{O} \models K \sqsubseteq M$. Consider an arbitrary literal $L \in K$; if $L \in \mathsf{core}(v)$, then $\top \sqsubseteq L$ is added to $\mathcal{S}(v)$ in step 1, and if $L \in K \setminus \mathsf{core}(v)$, then $L \sqsubseteq L$ is added to $\mathcal{S}(v)$ in step 2; in either case, we have that $K \sqsubseteq L \mathrel{\hat{\in}} \mathcal{S}(v)$ holds after step 2, and Lemma 12 ensures that this property is preserved during the algorithm's execution. But then, since v is complete for $K \sqsubseteq M$, we have $K \sqsubseteq M \mathrel{\hat{\in}} \mathcal{S}(v)$ by Theorem 6, as required. □

Finally, we determine the complexity of our algorithm.

Proposition 14 (Termination)**.** *When applied to $\mathcal{O}$, $\mathcal{Q}$, a context structure $\mathcal{D} = \langle \mathcal{V}, \mathcal{E}, \mathsf{core}, \prec \rangle$, an expansion strategy* strategy*, and a covering mapping ϑ, Algorithm 11 terminates in time polynomial in $4^{|\mathbf{L}|^2}$, $|\mathsf{strategy}|_{\mathbf{L}} + |\mathcal{V}|$, and $\|\mathcal{O}\| + \|\mathcal{Q}\|$.*

Proof. When applied to arguments over $\mathbf{L}$, the result of the expansion strategy is clearly over $\mathbf{L}$ as well; therefore, the context structure and the clause system computed using Algorithm 11 are over $\mathbf{L}$. The number of different clauses over $\mathbf{L}$ is bounded by $c = 2^{|\mathbf{L}|} \cdot 2^{|\mathbf{L}|} = 4^{|\mathbf{L}|}$ since each literal from $\mathbf{L}$ can independently occur in the clause antecedent and/or the clause consequent. Algorithm 11 is applied to a context structure with $|\mathcal{V}|$ contexts, and the expansion strategy can introduce at most $|\mathsf{strategy}|_{\mathbf{L}}$ additional contexts; hence, the total number of contexts introduced in the algorithm is bounded by $m = |\mathsf{strategy}|_{\mathbf{L}} + |\mathcal{V}|$.

We call all objects needed to apply an inference rule from Table 3 *premises*; for the Succ rule, this includes set $\mathbf{B}_p$. We next show that, for the Hyper, Pred, and Succ rules, the number of different premises is polynomial in $c^{|\mathbf{L}|}$, m, and $\|\mathcal{O}\| + \|\mathcal{Q}\|$; recall that $|\mathbf{L}| \leq \|\mathcal{O}\| + \|\mathcal{Q}\|$. For the Hyper rule, context $v \in \mathcal{V}$ can be chosen in at most m ways, clause $\bigsqcap_i A_i \sqsubseteq M \in \mathcal{O}$ can be chosen in at most $|\mathcal{O}|$ ways, and each of the $n \leq |\mathbf{L}|$ clauses $K_i \sqsubseteq M_i \sqcup A_i \in \mathcal{S}(v)$ can be chosen in at most c ways, so there are at most $c^{|\mathbf{L}|}$ ways of choosing such a set of clauses. For the Pred rule, edge $\langle v, u, \exists R.A \rangle \in \mathcal{E}$ can be chosen in at most $m^2 \cdot |\mathbf{L}|$ ways, clause $A \sqcap \bigsqcap_i B_i \sqsubseteq \bigsqcup_j \forall \mathsf{inv}(R).C_j \in \mathcal{S}(u)$ or $\bigsqcap_i B_i \sqsubseteq \bigsqcup_j \forall \mathsf{inv}(R).C_j \in \mathcal{S}(u)$ can be chosen in at most c ways, and each of the $n+1 \leq |\mathbf{L}|$ clauses $K_0 \sqsubseteq M_0 \sqcup \exists R.A \in \mathcal{S}(v)$ and $K_i \sqsubseteq M_i \sqcup \forall R.B_i \in \mathcal{S}(v)$ can be chosen in at most c ways, so there are at most $c^{|\mathbf{L}|}$ ways of choosing such a set of clauses. For the Succ rule, context $v \in \mathcal{V}$ can be chosen in at most m ways, literal $\exists R.A$ can be chosen in at most $|\mathbf{L}|$ ways, clause $K \sqsubseteq M \sqcup \exists R.A \in \mathcal{S}(v)$ can be chosen in at most c ways, and there are at most $2^{|\mathbf{L}|}$ different sets $\mathbf{B}_p$.

Each rule in Table 3 adds a clause $K \sqsubseteq M$ to some $\mathcal{S}(v)$ only if $K \sqsubseteq M \mathrel{\hat{\notin}} \mathcal{S}(v)$; thus, due to Lemma 12, each such $K \sqsubseteq M$ can be added to some $\mathcal{S}(v)$ at most once; but then, the Elim rule can eliminate such $K \sqsubseteq M$ from $\mathcal{S}(v)$ at most once as well. Thus, no inference rule is applied twice to the same premises, so the number of inference rule applications is bounded by a number that is polynomial in $c^{|\mathbf{L}|}$, m, and $\|\mathcal{O}\| + \|\mathcal{Q}\|$. Moreover, strategy is computable in polynomial time, so each inference rule can be applied to fixed premises in polynomial time. Consequently, the algorithm can be implemented so that it runs in time polynomial in $c^{|\mathbf{L}|}$, m, and $\|\mathcal{O}\| + \|\mathcal{Q}\|$. □

3.3. Redundancy Elimination

Resolution can often derive *redundant* clauses—that is, clauses that are not necessary for a proof. Such clauses can give rise to a large number of other redundant clauses, so it is beneficial to detect and eliminate redundant clauses whenever possible. To

Table 3: Consequence-based inference rules for $\mathcal{ALCI}$

Hyper	If	$\bigsqcap_{i=1}^{n} A_i \sqsubseteq M \in \mathcal{O}$,
		$K_i \sqsubseteq M_i \sqcup A_i \in \mathcal{S}(v)$ with $A_i \not\prec_v M_i$ for $1 \leq i \leq n$,
		and $\bigsqcap_{i=1}^{n} K_i \sqsubseteq M \sqcup \bigsqcup_{i=1}^{n} M_i \mathrel{\hat{\notin}} \mathcal{S}(v)$,
	then	add $\bigsqcap_{i=1}^{n} K_i \sqsubseteq M \sqcup \bigsqcup_{i=1}^{n} M_i$ to $\mathcal{S}(v)$.
Pred	If	$\langle v, u, \exists R.A \rangle \in \mathcal{E}$,
		$A \sqcap \bigsqcap_{i=1}^{n} B_i \sqsubseteq \bigsqcup_{j=1}^{m} \forall \mathsf{inv}(R).C_j \in \mathcal{S}(u)$ or $\bigsqcap_{i=1}^{n} B_i \sqsubseteq \bigsqcup_{j=1}^{m} \forall \mathsf{inv}(R).C_j \in \mathcal{S}(u)$,
		$K_0 \sqsubseteq M_0 \sqcup \exists R.A \in \mathcal{S}(v)$ with $\exists R.A \not\prec_v M_0$,
		$K_i \sqsubseteq M_i \sqcup \forall R.B_i \in \mathcal{S}(v)$ with $\forall R.B_i \not\prec_v M_i$ for $1 \leq i \leq n$,
		and $\bigsqcap_{i=0}^{n} K_i \sqsubseteq \bigsqcup_{i=0}^{n} M_i \sqcup \bigsqcup_{j=1}^{m} C_j \mathrel{\hat{\notin}} \mathcal{S}(v)$,
	then	add $\bigsqcap_{i=0}^{n} K_i \sqsubseteq \bigsqcup_{i=0}^{n} M_i \sqcup \bigsqcup_{j=1}^{m} C_j$ to $\mathcal{S}(v)$.
Succ	If	$K \sqsubseteq M \sqcup \exists R.A \in \mathcal{S}(v)$ with $\exists R.A \not\prec_v M$, and
		for $\mathbf{B}_p := \{B \mid K' \sqsubseteq M' \sqcup \forall R.B \in \mathcal{S}(v)$ and $\forall R.B \not\prec_v M'\}$,
		no edge $\langle v, u, \exists R.A \rangle \in \mathcal{E}$ exists such that $L \sqsubseteq L \mathrel{\hat{\in}} \mathcal{S}(u)$ for each $L \in \{A\} \cup \mathbf{B}_p$,
	then	let $\langle u, \mathsf{core}', \prec' \rangle := \mathsf{strategy}(\exists R.A, \mathbf{B}_k, \mathcal{D})$ where $\mathbf{B}_k := \{B \mid \top \sqsubseteq \forall R.B \in \mathcal{S}(v)\}$;
		if $u \in \mathcal{V}$, then let $\prec_u := \prec_u \cap \prec'$, and
		otherwise let $\mathcal{V} := \mathcal{V} \cup \{u\}$, $\mathsf{core}(u) := \mathsf{core}'$, $\prec_u := \prec'$, and $\mathcal{S}(u) := \{\top \sqsubseteq L \mid L \in \mathsf{core}(u)\}$;
		add $\langle v, u, \exists R.A \rangle$ to $\mathcal{E}$; and
		for each $L \in [\{A\} \cup \mathbf{B}_p] \setminus \mathsf{core}(v)$ such that $L \sqsubseteq L \mathrel{\hat{\notin}} \mathcal{S}(u)$, add $L \sqsubseteq L$ to $\mathcal{S}(u)$.
Elim	If	$K_1 \sqsubseteq M_1 \in \mathcal{S}(v)$,
		$K_2 \sqsubseteq M_2 \in \mathcal{S}(v)$, and
		$K_1 \sqsubseteq M_1$ is a strengthening of $K_2 \sqsubseteq M_2$ and the two clauses are distinct,
	then	remove $K_2 \sqsubseteq M_2$ from $\mathcal{S}(v)$.

this end, modern first-order theorem provers employ a number of *redundancy elimination rules* that eliminate or simplify certain clauses; Weidenbach [35] presents an overview of these techniques. Inspired by redundancy elimination in first-order theorem provers, we equipped our consequence-based framework with analogous possibilities, which we discuss in this section. Please note that redundancy elimination is optional: it may optimize the theorem proving process but is not needed for completeness. Similarly, the complexity of the algorithms that we present in Sections 4 and 5 is independent from any redundancy elimination techniques, but redundancy elimination can still improve the performance of these algorithms in practice.

First, the completeness Theorem 6 only requires each set of clauses $\mathcal{S}(v)$ to contain a strengthening of each conclusion obtained by applying the inference rules from Table 3; therefore, our inference rules derive a conclusion in $\mathcal{S}(v)$ only if $\mathcal{S}(v)$ does not already contain a strengthening of the conclusion. This is analogous to *forward subsumption* in first-order theorem proving: a newly derived clause is kept only if the clause is not redundant given the clauses derived thus far.

Second, the Elim rule deletes a clause $K_2 \sqsubseteq M_2$ from $\mathcal{S}(v)$ if $\mathcal{S}(v)$ contains a clause $K_1 \sqsubseteq M_1$ such that $K_1 \sqsubseteq M_1$ is a strengthening of $K_2 \sqsubseteq M_2$ and the two clauses are distinct. Note that $K_2 \sqsubseteq M_2$ can be derived before $K_1 \sqsubseteq M_1$, in which case the technique described in the previous paragraph will not eliminate $K_2 \sqsubseteq M_2$, and so $K_2 \sqsubseteq M_2$ can potentially participate in further redundant inferences. The Elim rule is thus analogous to *backward subsumption* in first-order theorem proving.

In the rest of this section we discuss certain aspects of our redundancy elimination techniques. We first consider *syntactic tautologies*, which are clauses of the form $K \sqcap L \sqsubseteq M \sqcup L$ with either K or M (or both) nonempty; note that this definition excludes clauses of the form $L \sqsubseteq L$, which are needed for completeness in Theorem 6 and are thus not redundant. The following proposition straightforwardly implies that our algorithm never derives syntactic tautologies; in practice this means that, whenever a possible consequence of the inference rule is a syntactic tautology, one can eagerly eliminate the clause (without checking whether the target clause set contains a strengthening of the consequence clause).

Proposition 15. *If, at some point in the execution of Algorithm 11, $K \sqsubseteq M \mathrel{\hat{\in}} \mathcal{S}(v)$ holds for some clause $K \sqsubseteq M$ and context v, then $L \sqsubseteq L \mathrel{\hat{\in}} \mathcal{S}(v)$ also holds at this point for each literal $L \in K$.*

Proof. The proof is by induction on the application of inference rules. Each clause introduced in steps 1 and 2 of Algorithm 11 or by the Succ rule obviously satisfies this property. Furthermore, an application of the Elim rule preserves this property by Lemma 12. Now consider an application of the Hyper or the Pred rule as shown in Table 3; then, for each literal L occurring in the antecedent of the conclusion, integer i exists such that $L \in K_i$; but then, the premise corresponding to K_i satisfies this property by the induction assumption, so $L \sqsubseteq L \mathrel{\hat{\in}} \mathcal{S}(v)$ holds. □

The Elim rule can also be used to define more powerful redundancy elimination rules. Assume that, for some context v, set $\mathcal{S}(v)$ contains clauses $K_1 \sqsubseteq M_1 \sqcup L$ and $K_2 \sqcap L \sqsubseteq M_2$ with $K_1 \subseteq K_2$ and $M_1 \subseteq M_2$. Clause $K_1 \sqcap K_2 \sqsubseteq M_1 \sqcup M_2$ is equal to $K_2 \sqsubseteq M_2$ and it is a logical consequence of $\mathcal{S}(v)$; now even though clause $K_2 \sqsubseteq M_2$ is not derived by an inference rule from Table 3 (in particular, note that the Hyper rule *does not* derive this clause), extending $\mathcal{S}(v)$ with this clause is sound. Thus, if we add $K_2 \sqsubseteq M_2$ to $\mathcal{S}(v)$, we can delete clause $K_2 \sqcap L \sqsubseteq M_2$ from $\mathcal{S}(v)$ using the Elim rule; essentially, we thus obtain a simplification rule that eliminates L from clause $K_2 \sqcap L \sqsubseteq M_2$. Analogously, if the clauses satisfy $K_2 \subseteq K_1$ and $M_2 \subseteq M_1$, then we can eliminate L from clause $K_1 \sqsubseteq M_1 \sqcup L$. This is closely related to the *contextual literal cutting* rule used in the first-order prover E [36].

Redundancy elimination may be difficult to implement fully in practice: even with complex index structures, checking whether some $\mathcal{S}(v)$ contains a strengthening of some clause may be inefficient. Please note, however, that both forms of redundancy elimination are *optional*—that is, they are not needed for completeness. Therefore, instead of $\hat{\in}$, one can use $\in$ to check rule applicability. Furthermore, one does not need to apply $\hat{\in}$ and the Elim rule exhaustively; for example, one can restrict redundancy checking to clauses of the form $\top \sqsubseteq A$ and $L \sqsubseteq L$ and index the latter using a hash table. Finally, an advanced implementation can "switch on" redundancy elimination only if set $\mathcal{S}(v)$ grows beyond a certain size. In most cases, however, eliminating syntactic tautologies should not cause any noticeable overhead.

3.4. Instantiating the Consequence-Based Framework

We next discuss ways of instantiating our consequence-based framework presented in Section 3.2. In particular, in Section 3.4.1 we discuss possibilities for initializing the context structure and the covering mapping that are given to Algorithm 11. Then, in Section 3.4.2 we discuss possible expansion strategies.

3.4.1. Initializing the Context Structure and the Covering Mapping

Algorithm 11 takes as input a context structure $\mathcal{D}$ without any edges and a covering mapping ϑ, and so the initialization of $\mathcal{D}$ and ϑ is deferred to the algorithm's users. We next discuss several reasonable approaches to initialization.

The first possibility is to introduce just one trivial (cf. Definition 4) context v_0. In such a case, step 2 of Algorithm 11 initializes $\mathcal{S}(v_0)$ with many clauses $L \sqsubseteq L$; this may give rise to a large number of inferences at context v_0, and may be further exacerbated by the fact that $\prec_{v_0}$ is empty. Hence, such an approach is unlikely to be efficient apart from in very simple cases.

The second possibility is to introduce a separate context $v_q = \vartheta(q)$ for each query $q = K \sqsubseteq M \in \mathcal{Q}$, define $\mathsf{core}(v_q) := K$, and define $\prec_{v_q}$ such that M is $\prec_{v_q}$-minimal and the remaining literals are ordered arbitrarily. This has the opposite effect from the first approach: step 2 of Algorithm 11 initializes each set $\mathcal{S}(v_q)$ with the least possible number of clauses $L \sqsubseteq L$, and ordering $\prec_{v_q}$ is as fine-grained as possible, which maximally reduces the number of inferences at context v_q. The main drawback, however, is that the number of contexts needed is linear in the size of $\mathcal{Q}$.

The third possibility is to introduce a context v_K for each conjunction K occurring in the antecedent of some query in $\mathcal{Q}$, define $\mathsf{core}(v_K) := K$, and define $\prec_{v_K}$ so that each literal occurring in the consequent of some query in $\mathcal{Q}$ is $\prec_{v_K}$-minimal while all remaining literals are ordered arbitrarily. We thus strike a balance between the number of contexts and the inferences performed.

Description logic reasoners are often used to classify an ontology $\mathcal{O}$, in which case $\mathcal{Q}$ contains $A \sqsubseteq B$ for all pairs of atomic concepts A and B. The second possibility then requires a quadratic number of contexts, which can be prohibitively large even on ontologies of moderate size. In contrast, the third possibility requires all atomic concepts in $\mathcal{O}$ to be minimal, which essentially "turns off" most ordering constraints. To address the former problem in hypertableau-based reasoners, Glimm et al. [37] developed a classification algorithm that gathers information about known and possible subsumptions from the ABoxes encountered during the algorithm's execution in order to reduce the number of pairs of atomic concepts for which $\mathcal{O} \models A \sqsubseteq B$ needs to be checked. This algorithm can be adapted to the consequence-based framework as well. In particular, let $\mathcal{S}$ be a clause system produced by Algorithm 11 such that $\top \sqsubseteq \bot \mathrel{\hat{\notin}} \mathcal{S}(v)$ holds for each context v—that is, $\mathcal{O}$ is satisfiable. Then, if a context v exists such that $\mathsf{core}(v) \subseteq \{A\}$ and $A \sqsubseteq B \mathrel{\hat{\in}} \mathcal{S}(v)$, by the soundness of our algorithm we have that $\mathcal{O} \models A \sqsubseteq B$—in other words, subsumption $A \sqsubseteq B$ is *known*. Furthermore, Proposition 16, shown below, tells us how to identify subsumptions that are *not possible*. By tightly integrating the consequence-based algorithm with the appropriately modified classification algorithm, we can extend $\mathcal{Q}$ as needed, thus considerably reducing the size of this set. Furthermore, whenever the classification algorithm adds a query $A \sqsubseteq B$ to $\mathcal{Q}$, we have two options: we can introduce a fresh context $v_{A \sqsubseteq B}$; or we can introduce or reuse context v_A, keeping in mind that, whenever we reuse an existing context, we must refine the literal ordering $\prec_{v_A}$ so that B becomes $\prec_{v_A}$-minimal.

Proposition 16. *Let $\mathcal{S}$ be a clause system obtained by applying Algorithm 11 to a normalized $\mathcal{ALCI}$ ontology $\mathcal{O}$ and a finite set of queries $\mathcal{Q}$. Then, $\mathcal{O} \not\models A \sqsubseteq B$ holds for all atomic concepts A and B for which there exists a context v such that $A \sqsubseteq A \mathrel{\hat{\in}} \mathcal{S}(v)$, $A \sqsubseteq B \mathrel{\hat{\notin}} \mathcal{S}(v)$, and no clause of the form $K \sqsubseteq M \sqcup B \in \mathcal{S}(v)$ exists that satisfies $K \subseteq \{A\}$ and $B \not\prec_v M$.*

Proof. The proof of this proposition relies on the proof of Theorem 6 presented in Appendix A, the notion of pre-models introduced in Appendix A.1, and the symbol $\vdash$ introduced in Appendix A.4.

Let $\mathcal{S}$, A, B, and v be as specified in the proposition. Then, $A \sqsubseteq B \hat{\notin} \mathcal{S}(v)$ clearly implies $A \sqsubseteq \bot \hat{\notin} \mathcal{S}(v)$; moreover, $\bot$ is empty and therefore $\prec_v$-minimal; together with $A \sqsubseteq A \hat{\in} \mathcal{S}(v)$, these observations imply $v \nvdash A \sqsubseteq \bot$. Thus, the construction of a pre-model $I = \langle \Delta, E, J \rangle$ in Appendix A.4 introduces an element $\delta^v_{A \sqsubseteq \bot} \in \Delta$ whose literal interpretation $J(\delta^v_{A \sqsubseteq \bot})$ is constructed as specified in Appendix A.2. Now consider an arbitrary clause $K \sqsubseteq M \sqcup B \in \mathcal{S}(v)$: by our assumption, at least one of $K \subseteq \{A\}$ or $B \not\succ_v M$ does not hold; thus, the conditions of Lemma 64 are not satisfied, and so clause $K \sqsubseteq M \sqcup B$ is not productive. Since this holds for all clauses in $\mathcal{S}(v)$ with B in the consequent, we have $B \notin J(\delta^v_{A \sqsubseteq \bot})$. Furthermore, due to $A \sqsubseteq A \hat{\in} \mathcal{S}(v)$ and Lemma 65, we have $A \in J(\delta^v_{A \sqsubseteq \bot})$. Thus, we have $I \not\models A \sqsubseteq B$, and so $\mathcal{O} \not\models A \sqsubseteq B$ holds by Corollary 61 and Claim 72. □

3.4.2. Expansion Strategies

In this section we discuss reasonable expansion strategies. We use the ontology $\mathcal{O}$ and query q defined in the following example to demonstrate how the choice of strategy affects the inferences of our algorithm.

Example 17. *Let $\mathcal{O}$ be the ontology containing axioms* (31)–(38)*, and let $q = A \sqsubseteq E$. Clearly, $\mathcal{O} \models q$ holds due to* (31)–(33)*.*

$$\begin{array}{rlrlrl}
A \sqsubseteq \exists R.B & (31) & A \sqsubseteq \exists S.B & (34) & A \sqcap B \sqsubseteq \exists T.F_1 & (36) \\
A \sqsubseteq \forall R.C_1 & (32) & A \sqsubseteq D \sqcup \forall S.C_2 & (35) & C_1 \sqcap C_2 \sqsubseteq C & (37) \\
B \sqcap C_1 \sqsubseteq \forall R^-.E & (33) & & & B \sqcap C \sqsubseteq \exists T.F_2 & (38)
\end{array}$$

The simplest possibility is to always reuse the trivial context v_0; thus, we define $\mathsf{trivial}(\exists R.A, \mathbf{B}_k, \mathcal{D}) = \langle v_0, \emptyset, \emptyset \rangle$. This strategy makes most sense if the trivial context v_0 is also used for initialization. The inferences of our algorithm on $\mathcal{O}$ with the trivial strategy are shown in Figure 2. As one can see, the algorithm is prolific: inferences (47)–(57) are not needed in order to derive (46). Thus, such an algorithm resembles various resolution-based decision procedures for DLs [31].

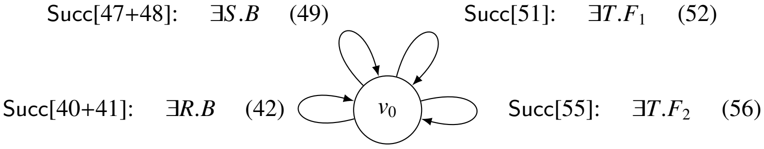
To reduce the number of irrelevant inferences, we can use a cautious approach and introduce a context v_A for each atomic concept A; thus, we define $\mathsf{cautious}(\exists R.A, \mathbf{B}_k, \mathcal{D}) = \langle v_A, \{A\}, \prec \rangle$, where $\prec$ is an arbitrary R-admissible ordering. The number of contexts is then limited to the number of atomic concepts occurring in $\mathcal{O}$ and $\mathcal{Q}$. Please remember that, whenever context v_A is reused, its literal ordering must be refined so that it stays admissible for each relevant role. As one can see in Figure 3, the cautious approach considerably reduces the inferences of our algorithm; for example, it does not derive clauses containing $\exists T.F_1$. Nevertheless, since we reuse context v_B to satisfy existential restrictions (59) and (66), the algorithm derives clause (70), which then leads to the derivation of (71) and the introduction of context v_{F_2}.

We can further reduce the number of irrelevant inferences by adopting an eager approach and introducing a context $v_{\mathbf{A}}$ for each set of atomic concepts $\mathbf{A}$; thus, we define $\mathsf{eager}(\exists R.A, \mathbf{B}_k, \mathcal{D}) = \langle v_{\{A\} \cup \mathbf{B}_k}, \{A\} \cup \mathbf{B}_k, \prec \rangle$, where $\prec$ is an arbitrary R-admissible ordering. As one can see in Figure 4, the algorithm now does not even derive clauses containing $\exists T.F_2$ and is considerably more efficient than in the previous two cases. In general, however, the eager strategy can be problematic because the number of introduced contexts can be exponential in the number of atomic concepts occurring in $\mathcal{O}$ and $\mathcal{Q}$.

One can refine all of the strategies discussed above by using a different context per role R. For example, the cautious strategy can be refined to use a context v^R_A for each atomic concept A and role R; thus, we define $\mathsf{cautious}_R(\exists R.A, \mathbf{B}_k, \mathcal{D}) = \langle v^R_A, \{A\}, \prec \rangle$, where $\prec$ is an arbitrary R-admissible ordering. A benefit of the refined cautious strategy is that context v^R_A is reused only to satisfy existential restrictions of the form $\exists R.A$, so the literal ordering used in the context never needs to be refined.

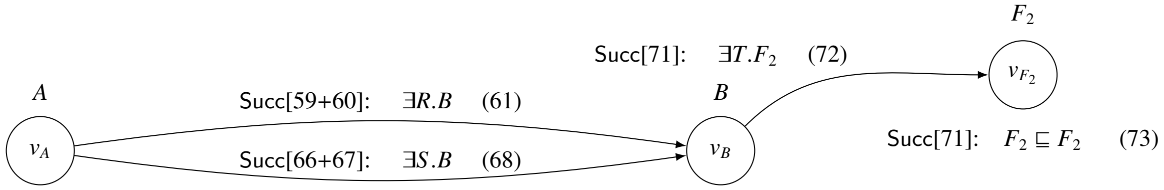
The cautious and the eager strategies appear to be most natural, striking a different balance between the number of contexts and the number of inferences in each context. In Section 8 we investigate the effectiveness of these strategies in practice. Our results suggest that, in most practical cases, the eager strategy may be most appropriate because it does not introduce too many contexts; however, we also identified cases in which the cautious strategy is more appropriate. This suggests that a hybrid approach may be useful: one uses the eager strategy until some predetermined number of contexts is introduced, after which one switches to the cautious strategy.

Although the eager strategy can, in general, introduce an exponential number of contexts, as we argue next, on $\mathcal{EL}$ and DL-Lite$_{horn}$ ontologies the eager strategy introduces only linearly many contexts. First, if $\mathcal{O}$ is an $\mathcal{EL}$ ontology, then all existential restrictions in $\mathcal{O}$ contain only atomic roles, and all universal restrictions in $\mathcal{O}$ contain only inverse roles; hence, the Succ rule is applicable only with $\mathbf{B}_k = \emptyset$, and so the eager strategy coincides with the cautious strategy. Thus, similarly to existing $\mathcal{EL}$ reasoning algorithms [23, 18], the eager strategy introduces only one context for each atomic concept. Second, if $\mathcal{O}$ is a DL-Lite$_{horn}$ ontology, then all universal restrictions occur in $\mathcal{O}$ only in axioms of the form $\top \sqsubseteq \forall R.B$; hence, if we apply the Hyper rule before the Succ rule, then all clauses $\top \sqsubseteq \forall R.A$ will be eagerly derived in each context, and so the Succ rule will be applicable to an existential restriction $\exists R.A$ only with $\mathbf{B}_k = \{B \mid \top \sqsubseteq \forall R.B \in \mathcal{O}\}$; since this set is uniquely identified by $\exists R.A$, the eager strategy will introduce at most one context for each existential restriction. Furthermore, if all existential restrictions occurring in $\mathcal{O}$ are of the form $\exists R.A_\top$ as discussed in Section 2.1, then, similarly to existing DL-Lite$_{horn}$ reasoning algorithms [19, 20], the eager strategy will introduce at most one context for each role occurring in an existential restriction.


Succ[47+48]: $\exists S.B$ (49)
Succ[51]: $\exists T.F_1$ (52)
Succ[40+41]: $\exists R.B$ (42)
v_0
Succ[55]: $\exists T.F_2$ (56)

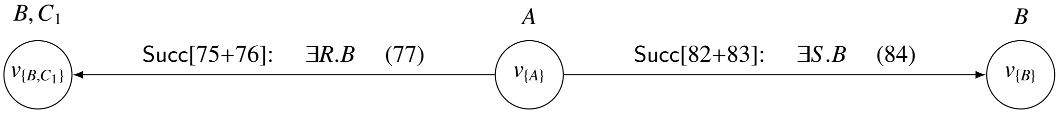
Initialization:	$A \sqsubseteq A$	(39)
Hyper[31+39]:	$A \sqsubseteq \exists R.B$	(40)
Hyper[32+39]:	$A \sqsubseteq \forall R.C_1$	(41)
Succ[40+41]:	$B \sqsubseteq B$	(43)
Succ[40+41]:	$C_1 \sqsubseteq C_1$	(44)
Hyper[33+43+44]:	$B \sqcap C_1 \sqsubseteq \forall R^-.E$	(45)
Pred[40+41+45]:	$A \sqsubseteq E$	(46)
Hyper[34+39]:	$A \sqsubseteq \exists S.B$	(47)
Hyper[35+39]:	$A \sqsubseteq D \sqcup \forall S.C_2$	(48)
Succ[47+48]:	$C_2 \sqsubseteq C_2$	(50)
Hyper[36+39+43]:	$A \sqcap B \sqsubseteq \exists T.F_1$	(51)
Succ[51]:	$F_1 \sqsubseteq F_1$	(53)
Hyper[37+44+50]:	$C_1 \sqcap C_2 \sqsubseteq C$	(54)
Hyper[38+43+44+50]:	$B \sqcap C_1 \sqcap C_2 \sqsubseteq \exists T.F_2$	(55)
Succ[55]:	$F_2 \sqsubseteq F_2$	(57)

Figure 2: Inferences with the Trivial Strategy


F_2
Succ[71]: $\exists T.F_2$ (72)
v_{F_2}
A
Succ[59+60]: $\exists R.B$ (61)
B
v_A
v_B
Succ[71]: $F_2 \sqsubseteq F_2$ (73)
Succ[66+67]: $\exists S.B$ (68)

Initialization:	$\top \sqsubseteq A$	(58)	Succ[59+60]:	$\top \sqsubseteq B$	(62)
Hyper[31+58]:	$\top \sqsubseteq \exists R.B$	(59)	Succ[59+60]:	$C_1 \sqsubseteq C_1$	(63)
Hyper[32+58]:	$\top \sqsubseteq \forall R.C_1$	(60)	Hyper[33+62+63]:	$C_1 \sqsubseteq \forall R^-.E$	(64)
Pred[59+60+64]:	$\top \sqsubseteq E$	(65)	Succ[66+67]:	$C_2 \sqsubseteq C_2$	(69)
Hyper[34+58]:	$\top \sqsubseteq \exists S.B$	(66)	Hyper[37+63+69]:	$C_1 \sqcap C_2 \sqsubseteq C$	(70)
Hyper[35+58]:	$\top \sqsubseteq D \sqcup \forall S.C_2$	(67)	Hyper[38+62+70]:	$C_1 \sqcap C_2 \sqsubseteq \exists T.F_2$	(71)

Figure 3: Inferences with the Cautious Strategy


B, C_1
A
B
$v_{\{B,C_1\}}$
Succ[75+76]: $\exists R.B$ (77)
$v_{\{A\}}$
Succ[82+83]: $\exists S.B$ (84)
$v_{\{B\}}$

Succ[75+76]:	$\top \sqsubseteq B$	(78)	Initialization:	$\top \sqsubseteq A$	(74)	Succ[75+83]:	$\top \sqsubseteq B$	(85)
Succ[75+76]:	$\top \sqsubseteq C_1$	(79)	Hyper[31+74]:	$\top \sqsubseteq \exists R.B$	(75)	Succ[82+83]:	$C_2 \sqsubseteq C_2$	(86)
Hyper[33+78+79]:	$\top \sqsubseteq \forall R^-.E$	(80)	Hyper[32+74]:	$\top \sqsubseteq \forall R.C_1$	(76)			
			Pred[75+80]:	$\top \sqsubseteq E$	(81)			
			Hyper[34+74]:	$\top \sqsubseteq \exists S.B$	(82)			
			Hyper[35+74]:	$\top \sqsubseteq D \sqcup \forall S.C_2$	(83)			

Figure 4: Inferences with the Eager Strategy

Finally, we relate the above strategies to existing consequence-based algorithms. Kazakov [21] considers only the eager strategy. Simančík et al. [22] present their algorithm with the eager strategy, but their "context partitioning" optimizations can be seen as providing more cautious strategies; moreover, their ConDOR reasoner implements only the refined cautious strategy.

4. Analyzing And-Branching Using ϵ-Free Decompositions

In the rest of this paper we develop our framework for a parametric analysis of the complexity of reasoning. As we mentioned in the introduction, and-branching and or-branching are two main sources of complexity [10]. In this section we focus on quantifying the effects of and-branching, whereas in Section 5 we turn our attention to or-branching.

4.1. Intuitions

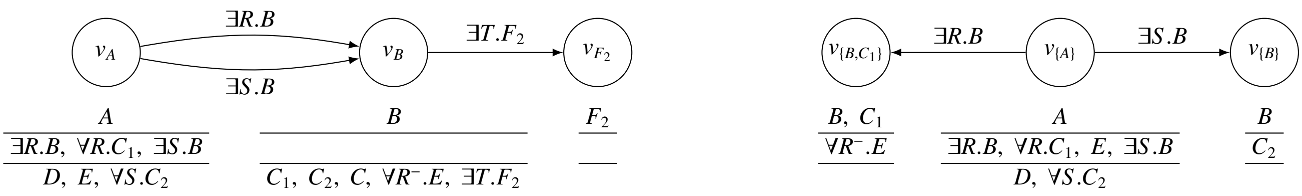
Let $\mathcal{O}$ and q be as specified in Example 17, and let $\mathbf{L}$ be the set of all literals occurring in $\mathcal{O}$ and q; note that $|\mathbf{L}| = 13$. Proposition 14 provides us with an estimate of the complexity of deciding $\mathcal{O} \models q$: we can introduce at most $2^{|\mathbf{L}|} = 2^{13}$ contexts using the eager strategy, and each set $\mathcal{S}(v)$ can contain at most $4^{|\mathbf{L}|} = 4^{13}$ clauses, so the algorithm runs in time exponential in $|\mathbf{L}| = 13$. This, however, is a crude estimate because both the number of contexts and the sizes of sets $\mathcal{S}(v)$ are vastly overestimated.

To obtain a better estimate of these values, in this section we extend the notion of a context structure to a *decomposition* of an ontology and a set of queries; since decompositions are proper extensions of context structures, one can apply our consequence-based algorithm to a decomposition by only slightly adapting the algorithm's initialization phase. The high-level principle behind our decompositions is similar to tree decompositions: both aim to identify sets of literals whose subsets can be relevant during reasoning. For example, given a vertex v in a tree decomposition, set $\mathcal{L}(v)$ determines the propositional variables that should be considered in "local" propositional interpretation associated with v, and so the number of such interpretations is bounded by $2^{|\mathcal{L}(v)|}$. Our notion of decomposition aims for a similar goal: each context in a decomposition will describe the clauses that the consequence-based algorithm can derive, which will then determine the worst-case complexity of the algorithm. Towards this goal, please remember that there are two sources of complexity in description logic reasoning [10]: existential and universal quantification cause and-branching, and disjunction causes or-branching. The latter corresponds to or-branching in propositional logic and can be characterized using tree decompositions; however, the former is unique to description logics and requires completely new techniques. Our decompositions therefore consist of two kinds of edges: $\exists R.A$-edges are defined as in context structures and capture information propagation due to and-branching, and ϵ-edges capture information propagation due to or-branching. In the rest of this section we focus on and-branching, so the decompositions we shall consider will be ϵ-*free* (i.e., they will not contain ϵ-edges). Please note that ϵ-free decompositions are completely different from and do not generalize tree decompositions. In Section 5 we then show how ϵ-edges can be used to capture or-branching.

As we discussed in Section 3.1, each context in a context structure describes one or more domain elements in a model of $\mathcal{O}$. A context structure can thus be seen as decomposing a description logic reasoning problem into several of interconnected propositional problems; given such a representation, the Hyper rule then locally solves each propositional subproblem, and the Pred and Succ rules handle the connections between subproblems. An ϵ-free decomposition $\mathcal{D}$ for $\mathcal{O}$ and q can in similar vein be understood as breaking up the problem of checking $\mathcal{O} \models q$ into several interconnected propositional problems, but with several differences. First, a context structure is constructed during reasoning, and so it cannot provide us with an a priori complexity estimate. In contrast, $\mathcal{D}$ is independent from any reasoning and contains "sufficiently many" contexts (i.e., all contexts needed by the Succ rule). Second, each context v in $\mathcal{D}$ is associated with three sets of literals $\mathsf{core}(v)$, $\mathsf{knw}(v)$, and $\mathsf{poss}(v)$ that describe the clauses one can derive in $\mathcal{S}(v)$. Set $\mathsf{core}(v)$ serves the same purpose as in context structures. Furthermore, set $\mathsf{knw}(v)$ contains the *known* literals—that is, for each literal $L \in \mathsf{knw}(v)$, our algorithm will derive $\top \sqsubseteq L$ in $\mathcal{S}(v)$. Finally, set $\mathsf{poss}(v)$ contains the *possible* literals—that is, each clause $K \sqsubseteq M \in \mathcal{S}(v)$ will satisfy $K \cup M \subseteq \mathsf{poss}(v)$. Clearly, all core literals are known, and all known literals are possible, so $\mathcal{D}$ should satisfy $\mathsf{core}(v) \subseteq \mathsf{knw}(v) \subseteq \mathsf{poss}(v)$ for each context v.

To ensure that each clause obtained by applying our consequence-based algorithm to $\mathcal{D}$ is a logical consequence of $\mathcal{O}$, we require $\mathcal{D}$ to be *sound*: in addition to the condition on $\exists R.A$-edges from Definition 8, the known literals in each context of $\mathcal{D}$ must logically follow from the context's core—that is, $\mathcal{O} \models \mathsf{core}(v) \sqsubseteq L$ should hold for each context v in $\mathcal{D}$ and each literal $L \in \mathsf{knw}(v)$. Furthermore, to ensure completeness of our algorithm, we require $\mathcal{D}$ to be *admissible*: roughly speaking, this notion requires $\mathcal{D}$ to contain all contexts that might be needed during our algorithm's execution, and sets $\mathsf{core}(v)$, $\mathsf{knw}(v)$, and $\mathsf{poss}(v)$ to correctly describe the types of clauses that our algorithm can derive in $\mathcal{S}(v)$. This will ensure that (i) the Succ rule is never applicable, and (ii) each clause in $\mathcal{S}(v)$ is of the form $\top \sqsubseteq L$ with $L \in \mathsf{knw}(v)$, or of the form $K \sqsubseteq M$ with $K \cup M \subseteq \mathsf{poss}(v) \setminus \mathsf{knw}(v)$. These observations allow us to capture the complexity of our algorithm using the following two parameters: decomposition *length* $\mathsf{ln}(\mathcal{D})$, defined as the number of contexts in $\mathcal{D}$, captures the combinatorial difficulty due to (i); and decomposition *width* $\mathsf{wd}(\mathcal{D})$, defined as the maximum cardinality of $\mathsf{poss}(v) \setminus \mathsf{knw}(v)$, captures the combinatorial difficulty due to (ii). We show that, when applied to $\mathcal{D}$, our consequence-based algorithm runs in time polynomial in $\mathsf{ln}(\mathcal{D})$ and $4^{\mathsf{wd}(\mathcal{D})^2}$; the latter factor is due to the fact that each set $\mathcal{S}(v)$ can contain at most $|\mathsf{knw}(v)| + 4^{|\mathsf{poss}(v) \setminus \mathsf{knw}(v)|}$ clauses.

An ontology and a set of queries can admit infinitely many sound, admissible, and ϵ-free decompositions, so a decomposition $\mathcal{D}$ with least $\mathsf{ln}(\mathcal{D})$ and $\mathsf{wd}(\mathcal{D})$ would provide us with a "general" difficulty of reasoning with a particular ontology and a set



(a) Cautious Strategy

(b) Eager Strategy

Figure 5: Examples of ϵ-Free Decompositions

of queries. This would be analogous to propositional satisfiability, where the treewidth of a propositional problem is defined as the smallest width over all tree decompositions of the problem. However, identifying such $\mathcal{D}$ is difficult for two reasons. First, checking semantic conditions in the definition of soundness requires reasoning, so verifying decomposition soundness can be as hard as the reasoning problem whose complexity we aim to analyze. Second, ontologies and queries exist for which all decompositions of minimal width have exponential length; we prove this claim for general (i.e., not only ϵ-free) decompositions, so we postpone the formal treatment of this property until Section 7. Hence, minimizing width can increase length and vice versa, which essentially prevents us from obtaining a "general" characterization of reasoning difficulty.

As a possible solution to these challenges, in Section 4.4 we present a practical algorithm that can construct in polynomial time an ϵ-free decomposition of $\mathcal{O}$ and $\mathcal{Q}$. To address the first issue, our algorithm approximates the entailment relation used in the definition of soundness. Thus, for some context v, set $\mathsf{knw}(v)$ approximates the set of literals entailed by $\mathsf{core}(v)$, and for each literal in $\mathsf{knw}(v)$ the relevant entailment can be established in polynomial time. To address the second issue, the algorithm is given a set of parameters $\mathcal{C}$ called a *control* that consists of an expansion strategy, an integer mln that imposes a bound on decomposition length, and two other parameters that determinize the decomposition construction process. For each $\mathcal{C}$, the ϵ-free decomposition of $\mathcal{O}$ and $\mathcal{Q}$ that is deterministically constructed by our algorithm is called the *$\mathcal{C}$-decomposition* of $\mathcal{O}$ and $\mathcal{Q}$. Parameter mln bounds the length of the $\mathcal{C}$-decomposition, which determines the decomposition width. Thus, the width and length of a $\mathcal{C}$-decomposition provide us with an account of the reasoning difficulty relative to $\mathcal{C}$.

To illustrate these ideas, Figure 5 shows two ϵ-free decompositions of $\mathcal{O}$ and q from Example 17 on page 15 obtained using the cautious and the eager expansion strategy. We use the same notation as for context structures, but show sets $\mathsf{core}(v)$, $\mathsf{knw}(v) \setminus \mathsf{core}(v)$, and $\mathsf{poss}(v) \setminus \mathsf{knw}(v)$ in a table below each context.

The ϵ-free decomposition $\mathcal{D}_1$ constructed from $\mathcal{O}$ and q using the cautions strategy is shown in Figure 5a. Since our goal is to check the entailment of $q = A \sqsubseteq E$, we introduce context v_A and add A to $\mathsf{core}(v_A)$, thus capturing the fact that the consequence-based algorithm initializes $\mathcal{S}(v_A)$ with (58). But then, from clauses (31), (32), and (34), the consequence-based algorithm can derive (59), (60), and (66); to reflect this, we add literals $\exists R.B$, $\forall R.C_1$, and $\exists S.B$ to $\mathsf{knw}(v_A)$. Furthermore, using clause (35), the consequence-based algorithm will derive (67); however, the consequent of the clause contains a disjunction, so neither of the literals is derived deterministically, and therefore we add D and $\forall S.C_2$ to $\mathsf{poss}(v_A)$. Next, we must satisfy existential restrictions $\exists R.B$ and $\exists S.B$; since we use the cautious strategy, we introduce a single context v_B, add the two edges from v_A to v_B, and initialize $\mathsf{core}(v_B)$ to B; furthermore, due to universal restrictions $\forall R.C_1 \in \mathsf{knw}(v_A) \subseteq \mathsf{poss}(v_A)$ and $\forall S.C_2 \in \mathsf{poss}(v_A)$, we add C_1 and C_2 to $\mathsf{poss}(v_B)$. Note that, due to the reuse of v_B, we cannot add C_1 to $\mathsf{knw}(v_B)$: not all domain elements represented by v_B necessarily satisfy C_1. We now continue our estimation for v_B: literal $\forall R^{-}.E$ is possible due to (33), concept C is possible due to (37), and literal $\exists T.F_2$ is possible due to (38). Moreover, since $\forall R^{-}.E \in \mathsf{poss}(v_B)$, literal E is possible for v_A. Note that the consequence-based algorithm derives (65); however, $\forall R^{-}.E$ is not in $\mathsf{knw}(v_B)$ due to context reuse, so we do not have sufficient information to add E to $\mathsf{knw}(v_A)$; in other words, $\mathsf{knw}(v_A)$ provides us only with a lower bound on "truly" known literals. Finally, we introduce context v_{F_2} to satisfy $\exists T.F_2 \in \mathsf{poss}(v_B)$ and thus obtain an admissible ϵ-free decomposition. Clearly, $\mathsf{wd}(\mathcal{D}_1) = 5$ and $\mathsf{ln}(\mathcal{D}_1) = 3$ so, when applied to $\mathcal{D}_1$, our inference rules can derive $3 \cdot 4^5$ clauses, which is a much more accurate estimate.

The ϵ-free decomposition $\mathcal{D}_2$ constructed from $\mathcal{O}$ and q using the eager strategy is shown in Figure 5b. The main difference compared to the previous case is that, due to the eager strategy, we now use distinct contexts, $v_{\{B,C_1\}}$ and $v_{\{B\}}$, to satisfy existential restrictions $\exists R.B$ and $\exists S.B$ at context $v_{\{A\}}$. Since literal $\forall R.C_1$ is known at $v_{\{A\}}$, we can now add C_1 to the core of $v_{\{B,C_1\}}$, which in turn allows us to make E known at $v_{\{A\}}$. Furthermore, literal $\exists T.F_2$ is now not possible at $v_{\{B\}}$. In this case, we have $\mathsf{wd}(\mathcal{D}_2) = 2$ and $\mathsf{ln}(\mathcal{D}_1) = 3$ so, when applied to $\mathcal{D}_2$, our inference rules can derive at most $3 \cdot 4^2$ clauses; thus, we have obtained an even better estimate of the running time of our consequence-based algorithm.

Reasoning with Horn-$\mathcal{ALCI}$ ontologies is generally easier in practice: such ontologies admit a single canonical model, which eliminates all or-branching; however, due to and-branching, reasoning is still $\mathsf{ExpTime}$-hard [17]. This is reflected in our notions of decompositions. In particular, if control $\mathcal{C}$ uses the eager expansion strategy and allows for "sufficiently many" contexts, the $\mathcal{C}$-decomposition $\mathcal{D}$ of a Horn ontology $\mathcal{O}$ has zero width. Then, all possible literals in $\mathcal{D}$ are also known, no or-branching is required, and so $\mathcal{D}$ contains a complete solution to the reasoning problem. By reducing the maximum number of contexts in $\mathcal{C}$, however, we can "trade off" and-branching for or-branching. Then, $\mathcal{D}$ does not necessarily have zero width, so further reasoning is needed; moreover, when applied to $\mathcal{D}$, our consequence-based algorithm derives only Horn clauses, and the size of the antecedent of each clause is bounded by $\mathsf{wd}(\mathcal{D})$. Furthermore, as we discussed in Section 3.4.2, if $\mathcal{O}$ is an $\mathcal{EL}$ or a DL-Lite$_{horn}$ ontology, the eager strategy introduces only linearly many contexts. Therefore, if $\mathcal{C}$ uses the eager strategy, the $\mathcal{C}$-decomposition of $\mathcal{O}$ has zero width and linear length, thus explaining the tractability of subsumption reasoning in $\mathcal{EL}$ and DL-Lite$_{horn}$.

It is instructive to consider the case when $\mathcal{O}$ and $\mathcal{Q}$ contain only propositional clauses—that is, if all literals in $\mathcal{O}$ and $\mathcal{Q}$ are atomic concepts. Then, for each admissible ϵ-decomposition $\mathcal{D}$ of $\mathcal{O}$ and $\mathcal{Q}$, the $\exists R.A$-edges are not needed and can be removed, but the decomposition may still contain multiple contexts that are used to (dis)prove entailment of different queries in $\mathcal{Q}$. Thus, only the Hyper rule is used when the consequence-based calculus is applied to $\mathcal{D}$, which essentially amount to solving the propositional problem(s) using hyperresolution. However, since $\mathsf{knw}(v)$ contains literals that deterministically follow from $\mathcal{O}$ for each context v, hyperresolution is restricted to the unknown literals.

Please note that neither redundancy elimination not literal orderings are needed to obtain our complexity results: the results hold even if each context is associated with the empty literal ordering, and if no redundant clauses are eliminated. We leave a study of whether redundancy elimination and literal orderings can provide us with a tighter complexity bound for future work.

4.2. Formalization

Throughout this section we fix a normalized $\mathcal{ALCI}$ ontology $\mathcal{O}$ and a finite set of queries $\mathcal{Q}$; furthermore, we let $\mathbf{L}$ be the set of literals that occur in $\mathcal{O} \cup \mathcal{Q}$, and we let ϵ be a special symbol not occurring in $\mathbf{L}$. We next formalize the notion of decompositions and then generalize the notion of context structure soundness to decompositions. In order to simplify the presentation, Definitions 18 and 19 both consider ϵ-edges that are used only in Section 5 for the analysis of or-branching; however, in this section we subsequently consider only decompositions without ϵ-edges.

Definition 18. *A* decomposition *of $\mathcal{O}$ and $\mathcal{Q}$ is a tuple $\mathcal{D} = \langle \mathcal{V}, \mathcal{E}, \mathsf{core}, \mathsf{knw}, \mathsf{poss}, \prec, \vartheta \rangle$ where $\mathcal{V} \subseteq \mathbf{X}$ is a finite set of contexts, set $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V} \times (\mathbf{L}^{\exists} \cup \{\epsilon\})$ contains edges labeled by existential restrictions (*$\exists R.A$-edges*) or by ϵ (*ϵ-edges*), functions* $\mathsf{core}, \mathsf{knw}, \mathsf{poss} : \mathcal{V} \to 2^{\mathbf{L}}$ *label contexts with sets of literals, function $\prec$ assigns to each context $v \in \mathcal{V}$ a literal ordering $\prec_v$, and function $\vartheta : \mathcal{Q} \to \mathcal{V}$ maps queries to contexts. For each vertex $v \in \mathcal{V}$, let $\mathsf{uknw}(v) := \mathsf{poss}(v) \setminus \mathsf{knw}(v)$. The* length *of $\mathcal{D}$ is* $\mathsf{ln}(\mathcal{D}) := |\mathcal{V}|$, *and the* width *of $\mathcal{D}$ is* $\mathsf{wd}(\mathcal{D}) := \max_{v \in \mathcal{V}} |\mathsf{uknw}(v)|$. *Decomposition $\mathcal{D}$ is* ϵ-free *if it contains no ϵ-edges.*

Definition 19. *A decomposition $\mathcal{D} = \langle \mathcal{V}, \mathcal{E}, \mathsf{core}, \mathsf{knw}, \mathsf{poss}, \prec, \vartheta \rangle$ of $\mathcal{O}$ and $\mathcal{Q}$ is* sound *if it satisfies all of the following conditions.*

(N1) $\mathcal{O} \models \mathsf{core}(v) \sqcap \exists R.A \sqsubseteq \exists R.[\mathsf{core}(u) \sqcap A]$ *for each $\exists R.A$-edge $\langle v, u, \exists R.A \rangle \in \mathcal{E}$.*

(N2) $\mathsf{core}(v) = \mathsf{core}(u)$ *for each ϵ-edge $\langle v, u, \epsilon \rangle \in \mathcal{E}$.*

(N3) $\mathcal{O} \models \mathsf{core}(v) \sqsubseteq L$ *for each context $v \in \mathcal{V}$ and each $L \in \mathsf{knw}(v)$.*

Condition (N1) of Definition 19 is the same as in Definition 8; condition (N2) is relevant only for decompositions that are not ϵ-free and will be explained in Section 5; and condition (N3) says that known literals should hold in all domain elements represented by a specific context. Please note that condition (N3) imposes only an upper bound on the known literals; thus, $\mathsf{knw}(v)$ can be obtained using a sound, but incomplete technique such as the one we present in Section 4.4.

Next, we extend the notion of context structure admissibility to decompositions. The ordering condition is the same as in context structures. Furthermore, to encapsulate all relevant parameters for the consequence-based algorithm, a decomposition contains a covering mapping, and the covering condition restates Definition 10. The ontology condition says that, if $\mathcal{O}$ contains a clause $K \sqsubseteq M$ and all of the literals in K are possible at context v, then the Hyper rule can derive M and so all literals in M should be possible at v too. Finally, the structural condition (S1) captures the intuitive relationship between the core, known, and possible literals; condition (S2) ensures that a decomposition correctly estimates all consequences of the Pred rule; and condition (S3) ensures that a decomposition contains sufficiently many contexts so that the Succ rule is never applicable.

Definition 20. *Let $\mathcal{D} = \langle \mathcal{V}, \mathcal{E}, \mathsf{core}, \mathsf{knw}, \mathsf{poss}, \prec, \vartheta \rangle$ be an ϵ-free decomposition of $\mathcal{O}$ and $\mathcal{Q}$. Let $v \in \mathcal{V}$ be an arbitrary context of $\mathcal{D}$, and let $K \sqsubseteq M$ be an arbitrary query; then, v is* sound *for $K \sqsubseteq M$ if* $\mathsf{core}(v) \subseteq K$; *$v$ is* complete *for $K \sqsubseteq M$ if $K \subseteq \mathsf{poss}(v)$ and M is $\prec_v$-minimal; and v* covers *$K \sqsubseteq M$ if v is both sound and complete for $K \sqsubseteq M$. Finally, $\mathcal{D}$ is* admissible *if all of the following conditions are satisfied.*

- *Structural conditions:*

  (S1) For each context $v \in \mathcal{V}$, we have $\mathsf{core}(v) \subseteq \mathsf{knw}(v) \subseteq \mathsf{poss}(v)$.

  (S2) For each edge $\langle v, u, \exists R.A \rangle \in \mathcal{E}$ and each literal $\forall \mathsf{inv}(R).C \in \mathsf{poss}(u)$, we have $C \in \mathsf{poss}(v)$.

  (S3) For each literal $\exists R.A \in \mathsf{poss}(v)$, an edge $\langle v, u, \exists R.A \rangle \in \mathcal{E}$ exists such that $\mathsf{poss}(u) \supseteq \{A\} \cup \{B \mid \forall R.B \in \mathsf{poss}(v)\}$.

- *Ordering condition: For each edge $\langle v, u, \exists R.A \rangle \in \mathcal{E}$, the literal ordering $\prec_u$ is R-admissible.*

- *Ontology condition: For each context $v \in \mathcal{V}$ and each clause $K \sqsubseteq M \in \mathcal{O}$ with $K \subseteq \mathsf{poss}(v)$, we have $M \subseteq \mathsf{poss}(v)$.*
- *Covering condition: Each query $q \in \mathcal{Q}$ is covered in the context $\vartheta(q)$.*

We can apply the consequence-based algorithm to a sound, admissible, and ϵ-free decomposition $\mathcal{D}$ as specified in Algorithm 21. Step 1 initializes each $\mathcal{S}(v)$ with a clause $\top \sqsubseteq L$ for each known literal $L \in \mathsf{knw}(v)$. This allows us to prove in Lemma 22 that the algorithm derives only clauses of the form $K \sqsubseteq M$ with $K \cup M \subseteq \mathsf{uknw}(v)$, which is central to our complexity argument in Proposition 24. Please note that, if step 1 were to consider only the literals in $\mathsf{core}(v)$, our algorithm would eventually derive a clause $\top \sqsubseteq L$ for each known literal $L \in \mathsf{knw}(v)$, but this might involve clauses containing combinations of literals in $\mathsf{knw}(v)$; thus, step 1 introduces derivation "shortcuts" for the known literals, which can improve the algorithm's performance in practice. Step 2 is analogous to the way the Succ rule initializes contexts, and it ensures that each atomic concept B that can be propagated to a context u through existential or universal quantification is relevant for u. Step 3 ensures that each query $q \in \mathcal{Q}$ is covered in context $\vartheta(q)$. Finally, step 4 applies exhaustively the consequence-based inference rules, but without the Succ rule. The latter is possible because $\mathcal{D}$ is admissible and thus contains all relevant contexts, which allows us to prove in Lemma 23 that the Succ rule is never applicable during the algorithm's execution. By combining this observation with the fact that $\mathcal{D}$ is sound, in Proposition 24 we show that our algorithm is sound and complete.

Algorithm 21. *The* ϵ-free decomposition-based reasoning algorithm *for $\mathcal{ALCI}$ takes a sound, admissible, and ϵ-free decomposition $\mathcal{D} = \langle \mathcal{V}, \mathcal{E}, \mathsf{core}, \mathsf{knw}, \mathsf{poss}, \prec, \vartheta \rangle$ of $\mathcal{O}$ and $\mathcal{Q}$. The algorithm (nondeterministically) computes a clause system $\mathcal{S}$ for $\mathcal{D}$ as follows.*

1. *Set $\mathcal{S}(v) := \{\top \sqsubseteq L \mid L \in \mathsf{knw}(v)\}$ for each context $v \in \mathcal{V}$.*
2. *For each edge $\langle v, u, \exists R.A \rangle \in \mathcal{E}$ and each concept $B \in \mathsf{uknw}(u)$ such that $B = A$ or $\forall R.B \in \mathsf{poss}(v)$, add $B \sqsubseteq B$ to $\mathcal{S}(u)$.*
3. *For each query $K \sqsubseteq M \in \mathcal{Q}$, let $v := \vartheta(K \sqsubseteq M)$; then, for each literal $L \in K \setminus \mathsf{knw}(u)$, add $L \sqsubseteq L$ to $\mathcal{S}(v)$.*
4. *Exhaustively apply rules* Hyper, Pred, *and* Elim *from Table 3.*

Lemma 22. *Let $\mathcal{S}$ be an arbitrary clause system encountered during step 4 of Algorithm 21, and let $v \in \mathcal{V}$ be an arbitrary context in $\mathcal{D}$. Then, each clause in $\mathcal{S}(v)$ is of the form $\top \sqsubseteq L$ with $L \in \mathsf{knw}(v)$, or $K \sqsubseteq M$ with $K \cup M \subseteq \mathsf{uknw}(v)$.*

Proof. Let $\mathcal{S}$ be a clause system at the beginning of step 4, and let $v \in \mathcal{V}$ be an arbitrary context. By Definition 20, if context v covers some query $K \sqsubseteq M$, then $K \subseteq \mathsf{poss}(v)$ holds; thus, each clause added to $\mathcal{S}(v)$ in steps 1–3 is clearly of the required form. We next show that this property is preserved in step 4.

Assume that the Hyper rule is applicable to some clause $\bigsqcap_{i=1}^{n} A_i \sqsubseteq M \in \mathcal{O}$. By the induction assumption, each premise from $\mathcal{S}(v)$ is of the form $K_i \sqsubseteq M_i \sqcup A_i$ with $K_i \cup M_i \subseteq \mathsf{uknw}(v)$ and $A_i \in \mathsf{poss}(v)$. Thus, $\sqcap_{i=1}^{n} A_i \subseteq \mathsf{poss}(v)$, so $M \subseteq \mathsf{poss}(v)$ by the ontology condition of Definition 20. The Hyper rule produces the clause $\bigsqcap_{i=1}^{n} K_i \sqsubseteq M \sqcup \bigsqcup_{i=1}^{n} M_i$. Now assume that there exists a literal $L \in M \cap \mathsf{knw}(v)$; then, $\top \sqsubseteq L \mathrel{\hat{\in}} \mathcal{S}(v)$ due to step 1 of Algorithm 21 and Lemma 12, which contradicts the assumption that the Hyper rule is applicable. Hence, $M \subseteq \mathsf{uknw}(v)$, and the produced clause is of the required form.

Assume that the Pred rule is applicable to an edge $\langle v, u, \exists R.A \rangle, \in \mathcal{E}$. By the induction assumption, the premises of the rule are of the form $K_0 \sqsubseteq M_0 \sqcup \exists R.A$ and $K_i \sqsubseteq M_i \sqcup \forall R.B_i$, where $K_i \cup M_i \subseteq \mathsf{uknw}(v)$ for each $0 \leq i \leq n$. The Pred rule produces the clause $\bigsqcap_{i=0}^{n} K_i \sqsubseteq \bigsqcup_{i=0}^{n} M_i \sqcup \bigsqcup_{j=1}^{m} C_j$. Now for each $1 \leq j \leq m$, by the induction assumption we have $\forall \mathsf{inv}(R).C_j \in \mathsf{poss}(u)$, so $C_j \in \mathsf{poss}(v)$ by condition (S2) of Definition 20. Furthermore, if $C_j \in \mathsf{knw}(v)$, then we have $\top \sqsubseteq C_j \mathrel{\hat{\in}} \mathcal{S}(v)$ due to step 1 of Algorithm 21 and Lemma 12, which contradicts the assumption that the Pred rule is applicable; consequently, we have $C_j \in \mathsf{uknw}(v)$. Hence, the clause produced by the rule is of the required form. □

Lemma 23. *The* Succ *rule is never applicable during step 4 of Algorithm 21.*

Proof. Let $\mathcal{S}$ be a clause system encountered in step 4 of Algorithm 21; let $v \in \mathcal{V}$ be a context; let $K \sqsubseteq M \sqcup \exists R.A \in \mathcal{S}(v)$ be a clause; and let $\mathbf{B}_p$ be as specified in Table 3. By Lemma 22, we have $\{B \mid \forall R.B \in \mathsf{poss}(v)\} \supseteq \mathbf{B}_p$ and $\exists R.A \in \mathsf{poss}(v)$. Thus, by condition (S3) of Definition 20, an edge $\langle v, u, \exists R.A \rangle \in \mathcal{E}$ exists with $\mathsf{poss}(u) \supseteq \{A\} \cup \{B \mid \forall R.B \in \mathsf{poss}(v)\} \supseteq \{A\} \cup \mathbf{B}_p$; furthermore, due to steps 1 and 2 of Algorithm 21 and Lemma 12, we have that $L \sqsubseteq L \mathrel{\hat{\in}} \mathcal{S}(u)$ holds for each $L \in \{A\} \cup \mathbf{B}_p$. Thus, the Succ rule is not applicable to context v and clause $K \sqsubseteq M \sqcup \exists R.A$ in $\mathcal{S}(v)$. □

Proposition 24. *Let $\mathcal{S}$ be a clause system obtained by applying Algorithm 21 to $\mathcal{O}$, $\mathcal{Q}$, and $\mathcal{D}$. Then, for each query $q \in \mathcal{Q}$, we have $\mathcal{O} \models q$ if and only if $q \mathrel{\hat{\in}} \mathcal{S}(\vartheta(q))$.*

Proof. Consider an arbitrary query $q = K \sqsubseteq M \mathrel{\hat{\in}} \mathcal{Q}$ and let $v = \vartheta(q)$; context v covers q by the covering condition of Definition 20, Decomposition $\mathcal{D}$ is sound, so, by condition (N3) of Definition 19, we have $\mathcal{O} \models \mathsf{core}(u) \sqsubseteq L$ for each clause $\top \sqsubseteq L$ added to some $\mathcal{S}(u)$ in step 1 of Algorithm 21; furthermore, clauses added in step 2 and 3 are tautologies; consequently, $\mathcal{S}$ is sound for $\mathcal{O}$. But then, since context v is sound for q, we have that $K \sqsubseteq M \mathrel{\hat{\in}} \mathcal{S}(v)$ implies $\mathcal{O} \models K \sqsubseteq M$ in the same way as in the proof of Proposition 13. Furthermore, for an arbitrary literal $L \in K$, either $\top \sqsubseteq L$ is added to $\mathcal{S}(v)$ in step 1, or $L \sqsubseteq L$ is added to $\mathcal{S}(v)$ in step 3; either way, we have $K \sqsubseteq L \mathrel{\hat{\in}} \mathcal{S}(v)$, and this property is preserved during the algorithm's execution by Lemma 12. Finally, Lemma 23 ensures that $\mathcal{S}$ is closed under the Hyper, Pred, and Succ rules; but then, since context v is complete for q, we have that $\mathcal{O} \models K \sqsubseteq M$ implies $K \sqsubseteq M \mathrel{\hat{\in}} \mathcal{S}(v)$ by Theorem 6. □

Proposition 25 provides us with our ultimate goal in this section: it shows that decomposition width and length provide us with a more accurate estimate of the complexity of subsumption reasoning.

Proposition 25 (Termination)**.** *When applied to some $\mathcal{O}$, $\mathcal{Q}$, and a sound, admissible, and ϵ-free decomposition $\mathcal{D}$ of $\mathcal{O}$ and $\mathcal{Q}$, Algorithm 21 terminates in time polynomial in $4^{\mathsf{wd}(\mathcal{D})^2}$, $\mathsf{ln}(\mathcal{D})$, and $\|\mathcal{O}\| + \|\mathcal{Q}\|$.*

Proof. Let $d = \mathsf{wd}(\mathcal{D})$ and $m = \mathsf{ln}(\mathcal{D})$, and recall that $|\mathbf{L}| \leq \|\mathcal{O}\| + \|\mathcal{Q}\|$. We next show that the number of inferences is polynomial in 4^{d^2}, m, and $\|\mathcal{O}\| + \|\mathcal{Q}\|$; this observation implies the claim of this proposition in the same way as in the proof of Proposition 14. By Lemma 22, for each context v, each clause in $\mathcal{S}(v)$ is of the form

- $\top \sqsubseteq L$ with $L \in \mathsf{knw}(v)$, and set $\mathcal{S}(v)$ can contain at most $|\mathbf{L}|$ such clauses, or
- $K \sqsubseteq M$ with $K \cup M \subseteq \mathsf{uknw}(v)$, and set $\mathcal{S}(v)$ can contain at most $c = 4^d$ such clauses.

For the Hyper rule, a context $v \in \mathcal{V}$ can be selected in at most m ways, and a clause $\bigsqcap_i A_i \sqsubseteq M \in \mathcal{O}$ can be selected in at most $|\mathcal{O}|$ ways. For each $A_i \in \mathsf{uknw}(v)$ there are at most c ways of choosing a matching clause $K_i \sqsubseteq M_i \sqcup A_i \in \mathcal{S}(v)$, and for each $A_i \in \mathsf{knw}(v)$ there is exactly one way of choosing a matching clause $\top \sqsubseteq A_i \in \mathcal{S}(v)$. Consequently, there are at most c^d ways of choosing the premises $K_i \sqsubseteq M_i \sqcup A_i \in \mathcal{S}(v)$.

For the Pred rule, there are at most $m^2 \cdot |\mathbf{L}|$ ways of choosing the edge $\langle v, u, \exists R.A \rangle \in \mathcal{E}$, at most $|\mathbf{L}| + 4^d$ ways of choosing the clause $A \sqcap \bigsqcap_i B_i \sqsubseteq \bigsqcup_j \forall \mathsf{inv}(R).C_j \in \mathcal{S}(u)$ or $\bigsqcap_i B_i \sqsubseteq \bigsqcup_j \forall \mathsf{inv}(R).C_j \in \mathcal{S}(u)$, and, analogously to the Hyper rule, c^d ways of choosing the premises $K_0 \sqsubseteq M_0 \sqcup \exists R.A \in \mathcal{S}(v)$ and $K_i \sqsubseteq M_i \sqcup \forall R.A \in \mathcal{S}(v)$. □

We would like to point out that the factor $4^{\mathsf{wd}(\mathcal{D})^2}$ in Proposition 25 is due to the fact that the Hyper and the Pred rules are based on hyperresolution: they fix one of the $4^{\mathsf{wd}(\mathcal{D})}$ clauses, and then try to find matches for each of the $\mathsf{wd}(\mathcal{D})$ literals in the antecedent, which gives rise to $4^{\mathsf{wd}(\mathcal{D})^2}$ combinations. It is, however, possible to develop a binary version of these inference rules, in which case the algorithm's complexity would be polynomial in $4^{\mathsf{wd}(\mathcal{D})}$. In our experience, hyperresolution tends to be more effective than binary resolution in practice, which is why we opted for this style of presentation.

4.3. ϵ-Free Decompositions and the Hypertableau Algorithm

In this section we show that ϵ-free decompositions explain to an extent the performance of the hypertableau algorithm and are thus not specific to consequence-based algorithms. Since the hypertableau algorithm cannot answer several queries at once, we consider only the case when set $\mathcal{Q}$ contains a single query of the form $A \sqsubseteq \bot$ for A an atomic concept—that is, when our goal is to determine the satisfiability of an atomic concept. The following theorem provides us with a key insight.

Theorem 26. *Let $\mathcal{O}$ be a normalized $\mathcal{ALCI}$ ontology, let $q = A \sqsubseteq \bot$, let D be a derivation of the hypertableau algorithm for $\mathcal{O}$ and the ABox $\{A(a)\}$, and let $\mathcal{D} = \langle \mathcal{V}, \mathcal{E}, \mathsf{core}, \mathsf{knw}, \mathsf{poss}, \prec, \vartheta \rangle$ be an admissible and ϵ-free decomposition of $\mathcal{O}$ and $\{q\}$. Then, for each ABox $\mathcal{A}_0$ labeling a vertex of D, there exists a mapping $\mu : \mathsf{ind}(\mathcal{A}_0) \to \mathcal{V}$ such that*

(a) if $L(s) \in \mathcal{A}_0$, then $L \in \mathsf{poss}(\mu(s))$, and

(b) if $R(s,t) \in \mathcal{A}_0$, then an atomic concept A exists such that $\langle \mu(s), \mu(t), \exists R.A \rangle \in \mathcal{E}$.

Proof. We prove the claim by induction of the depth of D, but we strengthen property (b) to the following property (b'):

(b') if $R(s,t) \in \mathcal{A}_0$, then some A exists such that $\langle \mu(s), \mu(t), \exists R.A \rangle \in \mathcal{E}$ and $\mathsf{poss}(\mu(t)) \supseteq \{A\} \cup \{B \mid \forall R.B \in \mathsf{poss}(\mu(s))\}$.

For the induction base, let $\mathcal{A}_0 = \{A(a)\}$ be the ABox labeling the root of D, and let $\mu(a) = \vartheta(q)$. Then $\mathcal{L}_{\mathcal{A}_0}(a) = \{A\} \subseteq \mathsf{poss}(\mu(a))$ holds by the covering condition from Definition 20, as required for property (a); moreover, property (b') holds vacuously. For the inductive step, let $\mathcal{A}_0$ be an ABox satisfying the property for some μ, and consider an ABox obtained from $\mathcal{A}_0$ as follows.

Assume that the Hyp-rule derives $L_j(s) \in \mathcal{A}_j$ from $\{A_1(s), \ldots, A_m(s)\} \subseteq \mathcal{A}_0$ and $A_1 \sqcap \ldots \sqcap A_m \sqsubseteq L_1 \sqcup \ldots \sqcup L_n \in \mathcal{O}$. By the induction assumption, we have $\{A_1, \ldots, A_m\} \subseteq \mathcal{L}_{\mathcal{A}_0}(s) \subseteq \mathsf{poss}(\mu(s))$; but then, since $\mathcal{D}$ satisfies the ontology condition from Definition 20, we have $L_j \in \mathsf{poss}(\mu(s))$, as required for property (a).

Assume that the $\exists$-rule derives $\{R(s,t),\ A(t)\} \subseteq \mathcal{A}_1$ from $\exists R.A(s) \in \mathcal{A}_0$. Then $\exists R.A \in \mathsf{poss}(\mu(s))$ holds by the induction assumption, so a context $u \in \mathcal{V}$ exists that satisfies condition (S3) from Definition 20. We extend μ by defining $\mu(t) := u$. ABox $\mathcal{A}_1$ and the extended mapping μ then clearly satisfy properties (a) and (b').

Assume that the $\forall^+$-rule derives $B(t) \in \mathcal{A}_1$ from $\{\forall R.B(s),\ R(s,t)\} \subseteq \mathcal{A}_0$. By the induction assumption, property (a) then implies $\forall R.B \in \mathsf{poss}(\mu(s))$, and property ($b'$) is satisfied for some atomic concept A, which immediately implies $B \in \mathsf{poss}(\mu(t))$, as required for property (a).

Assume that the $\forall^-$-rule derives $B(s) \in \mathcal{A}_1$ from $\{[\forall \mathsf{inv}(R).B](t),\ R(s,t)\} \subseteq \mathcal{A}_0$. By the induction assumption, property (a) then implies $\forall \mathsf{inv}(R).B \in \mathsf{poss}(\mu(t))$, and property (b') is satisfied for some atomic concept A. But then, condition (S2) from Definition 20 implies $B \in \mathsf{poss}(\mu(s))$, as required for property (a). □

Intuitively, Theorem 26 says that each ABox $\mathcal{A}_0$ labeling a derivation vertex can be "embedded" via some mapping μ into each admissible and ϵ-free decomposition $\mathcal{D}$ of $\mathcal{O}$ and $\mathcal{Q}$. Thus, for each individual $s \in \mathsf{ind}(\mathcal{A}_0)$, set $\mathsf{poss}(\mu(s))$ provides us with an upper bound on $\mathcal{L}_{\mathcal{A}_0}(s)$—a set used in the definition of blocking. Now let $\ell = \max_{v \in V} |\mathsf{poss}(v)|$; then, for each context in $\mathcal{D}$ we can have at most 2^ℓ different labels, so the total number of different labels is bounded by $\wp = \mathsf{ln}(\mathcal{D}) \cdot 2^\ell$; in other words, $\wp$ provides us with an upper bound on the number of nonblocked individuals in $\mathcal{A}_0$, which is analogous to Proposition 25. As we explained in Section 2.2, however, the number of indirectly blocked individuals can be exponentially larger due to dynamic blocking, but our decompositions do not analyze the computations caused by this effect.

Furthermore, assume that we modify the algorithm in two ways: (i) we explicitly maintain the mapping μ from Theorem 26, and (ii) whenever the $\exists$-rule introduces a new individual t, we immediately derive $L(t)$ for each literal $L \in \mathsf{knw}(\mu(t))$. Provided that $\mathcal{D}$ is sound, these changes do not affect the soundness of the algorithm, but they ensure that also $L(s) \in \mathcal{A}_0$ holds in Theorem 26 for each individual $s \in \mathsf{ind}(\mathcal{A}_0)$ and each literal $L \in \mathsf{knw}(\mu(s))$. Then, the number of different labels for each context is bounded by $2^{\mathsf{wd}(\mathcal{D})}$, so the total number of labels and the number of nonblocked individuals is bounded by $\mathsf{ln}(\mathcal{D}) \cdot 2^{\mathsf{wd}(\mathcal{D})}$. Please note that the standard hypertableau algorithm would eventually derive all literals introduced in (ii), but since the order in which derivation rules are applied is don't-care nondeterministic, this may occur only after a substantial amount of work. In contrast, modification (ii) eagerly introduces all known facts, which, through "earlier" blocking, can improve the algorithm's performance.

One can analogously use decompositions to obtain an automata-based algorithm running in time polynomial in $\mathsf{ln}(\mathcal{D})$ and $2^{\mathsf{wd}(\mathcal{D})}$, but we do not discuss the technical details any further for the sake of brevity.

4.4. Constructing ϵ-Free Decompositions

We now present our algorithm for computing ϵ-free decompositions that we outlined in Section 4.1. We first formalize the notion of a control, which encapsulates the parameters that guide our algorithm in the construction process. Please recall that $\mathbf{X}$ is the countably infinite set of all contexts that was fixed in Section 3.

Definition 27. *A* control *is a tuple* $\mathcal{C} = \langle \mathsf{mln}, \mathsf{initialization}, \mathsf{strategy}, \mathsf{precedence} \rangle$ *with the following components.*

- mln *is a positive integer.*
- $\mathsf{initialization}$ *is a polynomial-time computable function that takes a normalized $\mathcal{ALCI}$ ontology $\mathcal{O}$, a finite set of queries $\mathcal{Q}$, and a positive integer* mln. *The result of* $\mathsf{initialization}(\mathcal{O}, \mathcal{Q}, \mathsf{mln})$ *is a pair* $\langle \mathcal{D}, \vartheta \rangle$ *where*
  - *$\mathcal{D}$ is a context structure over the literals occurring in $\mathcal{O} \cup \mathcal{Q}$ such that $\mathcal{D}$ contains no edges,* $\mathsf{ln}(\mathcal{D}) \leq \mathsf{mln}$, *and $\mathcal{D}$ contains a trivial context, and*
  - *ϑ is a covering mapping from $\mathcal{Q}$ to $\mathcal{D}$.*
- $\mathsf{strategy}$ *is a polynomial-time computable function that takes an existential restriction $\exists R.A$, a set of atomic concepts $\mathbf{B}_k$, a decomposition $\mathcal{D} = \langle \mathcal{V}, \mathcal{E}, \mathsf{core}, \mathsf{knw}, \mathsf{poss}, \prec, \vartheta \rangle$ that contains a trivial context, and a positive integer* mln. *The result of* $\mathsf{strategy}(\exists R.A, \mathbf{B}_k, \mathcal{D}, \mathsf{mln})$ *is a triple* $\langle u, \mathsf{core}', \prec' \rangle$ *where*
  - core' *is a subset of* $\{A\} \cup \mathbf{B}_k$,
  - *$\prec'$ is an R-admissible literal ordering, and*
  - *either $u \in \mathbf{X} \setminus \mathcal{V}$ is a fresh context, or $u \in \mathcal{V}$ is a context in $\mathcal{D}$ such that* $\mathsf{core}(u) = \mathsf{core}'$.

  If $|\mathcal{V}| \geq \mathsf{mln}$, *then* $\mathsf{strategy}$ *must return a context $u \in \mathcal{V}$ from $\mathcal{D}$; since $\mathcal{D}$ contains a trivial context, a context satisfying the required conditions is guaranteed to exist.*
- $\mathsf{precedence}$ *is a strict total order on the set of all pairs of the form $\langle \exists R.A, v \rangle$ with $\exists R.A \in \Sigma_L^\exists$ and $v \in \mathbf{X}$.*

Intuitively, a control $\mathcal{C}$ specifies which contexts to introduce for a given ontology and a set of queries. In particular, function $\mathsf{initialization}$ specifies how to assign the queries to contexts; the function returns a context structure and a covering mapping that our algorithm will use as a starting point for the construction of an ϵ-free decomposition: the algorithm will immediately extend the given context structure into an ϵ-free decomposition by setting $\mathsf{poss}(v) := \mathsf{knw}(v) := \mathsf{core}(v)$ for each context v. Furthermore, function $\mathsf{strategy}$ is analogous to an expansion strategy and it specifies which contexts to introduce in order to satisfy existential and universal restrictions. Moreover, as we discussed in Section 4.1, the width and the length of a decomposition are not independent, and an exponential length may be needed to minimize the width. To overcome this problem, mln imposes an upper bound on decomposition length, and $\mathsf{initialization}$ and $\mathsf{strategy}$ are both required to honor this restriction. In particular, $\mathsf{strategy}$ is required to reuse a context whenever this bound has been exceeded; the decomposition being constructed will always contain the trivial context, so context reuse will always be possible. Finally, $\mathsf{precedence}$ determines a unique order in which fresh contexts are introduced, so that the resulting ϵ-free decomposition is uniquely determined by $\mathcal{C}$, the ontology, and the set of queries.

Table 4: ϵ-free decomposition construction rules

$\mathbf{H}_k$	If then	$K \subseteq \mathsf{knw}(v)$, $K \sqsubseteq L \in \mathcal{O}$, and $L \notin \mathsf{knw}(v)$, add L to $\mathsf{knw}(v)$.
$\mathbf{H}_p$	If then	$K \subseteq \mathsf{poss}(v)$, $K \sqsubseteq M \in \mathcal{O}$, and $M \nsubseteq \mathsf{poss}(v)$, add each literal in M to $\mathsf{poss}(v)$.
$\mathbf{P}_k$	If then	$\langle v, u, \exists R.A \rangle \in \mathcal{E}$, $\exists R.A \in \mathsf{knw}(v)$, $\forall \mathsf{inv}(R).C \in \mathsf{knw}(u)$, and $C \notin \mathsf{knw}(v)$, add C to $\mathsf{knw}(v)$.
$\mathbf{P}_p$	If then	$\langle v, u, \exists R.A \rangle \in \mathcal{E}$, $\forall \mathsf{inv}(R).C \in \mathsf{poss}(u)$, and $C \notin \mathsf{poss}(v)$, add C to $\mathsf{poss}(v)$.
$\mathbf{S}$	If then	$\exists R.A \in \mathsf{poss}(v)$, and for $\mathbf{B}_p = \{B \mid \forall R.B \in \mathsf{poss}(v)\}$, no edge $\langle v, u, \exists R.A \rangle \in \mathcal{E}$ exists such that $\mathsf{poss}(u) \supseteq \{A\} \cup \mathbf{B}_p$, let $\langle u, \mathsf{core}', \prec' \rangle := \mathsf{strategy}(\exists R.A, \mathbf{B}_k, \mathcal{D}, \mathsf{mln})$ where $\mathbf{B}_k = \{B \mid \forall R.B \in \mathsf{knw}(v)\}$; if $u \in \mathcal{V}$, then let $\prec_u := \prec_u \cap \prec'$, and otherwise let $\mathcal{V} := \mathcal{V} \cup \{u\}$, $\mathsf{core}(u) := \mathsf{knw}(u) := \mathsf{poss}(u) := \mathsf{core}'$, and $\prec_u := \prec'$; add $\langle v, u, \exists R.A \rangle$ to $\mathcal{E}$; and add each literal in $\{A\} \cup \mathbf{B}_p$ to $\mathsf{poss}(u)$.

Throughout the rest of section we fix a normalized $\mathcal{ALCI}$ ontology $\mathcal{O}$, a finite set of queries $\mathcal{Q}$, and a control $\mathcal{C}$; furthermore, we let $\mathbf{L}$ be the set of literals that occur in $\mathcal{O} \cup \mathcal{Q}$. We are now ready to present our decomposition construction algorithm.

Algorithm 28. *The* ϵ-free decomposition construction algorithm *takes a control* $\mathcal{C} = \langle \mathsf{mln}, \mathsf{initialization}, \mathsf{strategy}, \mathsf{precedence} \rangle$*,* $\mathcal{O}$*, and* $\mathcal{Q}$*, and it constructs an* ϵ*-free decomposition* $\mathcal{D} = \langle \mathcal{V}, \mathcal{E}, \mathsf{core}, \mathsf{knw}, \mathsf{poss}, \prec, \vartheta \rangle$ *of* $\mathcal{O}$ *and* $\mathcal{Q}$ *as follows.*

1. *Let* $\langle \mathcal{D}, \vartheta \rangle := \mathsf{initialization}(\mathcal{O}, \mathcal{Q}, \mathsf{mln})$ *and then extend* $\mathcal{D}$ *to the decomposition* $\mathcal{D} = \langle \mathcal{V}, \mathcal{E}, \mathsf{core}, \mathsf{knw}, \mathsf{poss}, \prec, \vartheta \rangle$ *by setting* $\mathsf{poss}(v) := \mathsf{knw}(v) := \mathsf{core}(v)$ *for each context* $v \in \mathcal{V}$*.*

2. *For each query* $K \sqsubseteq M \in \mathcal{Q}$*, let* $v := \vartheta(K \sqsubseteq M)$*; then, set* $\mathsf{poss}(v) := \mathsf{poss}(v) \cup K$*.*

3. *Exhaustively apply all rules from Table 4 apart from rule* **S***.*

4. *Stop and return* $\mathcal{D}$ *if rule* **S** *is not applicable.*

5. *Let* $\langle \exists R.A, v \rangle$ *be the minimal pair w.r.t.* $\mathsf{precedence}$ *such that rule* **S** *is applicable to* $\exists R.A$ *and* v*; then, apply rule* **S** *to* $\exists R.A$ *and* v*, and proceed to step 3.*

A decomposition produced by this algorithm is called a $\mathcal{C}$-decomposition *of* $\mathcal{O}$ *and* $\mathcal{Q}$*.*

Algorithm 28 can be intuitively understood as follows. Step 1 uses function $\mathsf{initialization}$ to introduce the contexts needed to cover the queries in $\mathcal{Q}$, and then it initializes the sets of known and possible literals for each context to be equal to the context's core. Next, for each query $q = K \sqsubseteq M \in \mathcal{Q}$ and each literal $L \in K$, step 2 of Algorithm 28 adds L to $\mathsf{poss}(\vartheta(q))$ in order to reflect the fact that step 3 of Algorithm 21 introduces a clause of the form $L \sqsubseteq L$ for each such L. Finally, in steps 3–5, the algorithm applies the rules shown in Table 4. Rules $\mathbf{H}_k$ and $\mathbf{H}_p$ correspond to the Hyper rule; rules $\mathbf{P}_k$ and $\mathbf{P}_p$ correspond to the Pred rule; and rule $\mathbf{S}$ corresponds to the Succ rule. Rules $\mathbf{H}_k$ and $\mathbf{P}_k$, however, are applied to the sets of known literals, and so they determine a lower bound on the known consequences of rules Hyper and Pred, respectively. In contrast, rules $\mathbf{H}_p$ and $\mathbf{P}_p$ are applied to the sets of possible literals, and so they determine an upper bound on the possible consequences of rules Hyper and Pred, respectively. Finally, rule $\mathbf{S}$ introduces all possible contexts that may be introduced by the Succ rule; the rule is applied with the lowest priority and in accordance with the precedence $\mathsf{precedence}$ from $\mathcal{C}$ in order to make the algorithm deterministic. In the rest of this section we prove the following theorem, which summarizes the formal properties of Algorithm 28.

Theorem 29. *When applied to* $\mathcal{O}$*,* $\mathcal{Q}$*, and* $\mathcal{C}$*, Algorithm 28 terminates in time polynomial in* mln *and* $\|\mathcal{O}\| + \|\mathcal{Q}\|$*. The result is a unique, sound, admissible, and* ϵ*-free decomposition* $\mathcal{D}$ *of* $\mathcal{O}$ *and* $\mathcal{Q}$ *satisfying* $\mathsf{ln}(\mathcal{D}) \leq \mathsf{mln}$*.*

Let $\mathcal{D} = \langle \mathcal{V}, \mathcal{E}, \mathsf{core}, \mathsf{knw}, \mathsf{poss}, \prec, \vartheta \rangle$ be a decomposition obtained by the decomposition construction algorithm. Clearly, $\mathcal{D}$ refers only to the literals occurring in $\mathcal{O} \cup \mathcal{Q}$, and it does not contain ϵ-edges; hence, $\mathcal{D}$ is an ϵ-free decomposition of $\mathcal{O}$ and $\mathcal{Q}$. We next prove the remaining claims of Theorem 29.

Claim 30. *The decomposition construction algorithm terminates in time polynomial in* mln *and* $\|\mathcal{O}\| + \|\mathcal{Q}\|$, *and the resulting decomposition* $\mathcal{D}$ *satisfies* $\mathsf{ln}(\mathcal{D}) \leq \mathsf{mln}$.

Proof. Since $\mathsf{initialization}$ is computable in polynomial time, steps 1 and 2 can be performed in time polynomial in mln and $\|\mathcal{O}\| + \|\mathcal{Q}\|$. By Definition 27, step 1 introduces at most mln contexts due to the requirement on $\mathsf{initialization}$, and rule **S** can introduce a new context in the call to $\mathsf{strategy}$ only if the total number of contexts is smaller than mln; thus, $|\mathcal{V}| \leq \mathsf{mln}$ clearly holds. Since $\mathcal{D}$ refers only to literals from $\mathbf{L}$, we have $|\mathcal{E}| \leq \mathsf{mln}^2 \cdot |\mathbf{L}|$ and, for each context $v \in \mathcal{V}$, we have

$$|\mathsf{core}(v)| \leq |\mathbf{L}|, \quad |\mathsf{knw}(v)| \leq |\mathbf{L}|, \quad \text{and} \quad |\mathsf{poss}(v)| \leq |\mathbf{L}|.$$

The decomposition construction algorithm is monotonic—that is, it never removes elements from the decomposition being constructed. Also, the precondition of each rule contains a negation of the rule's postcondition; thus, whenever a rule is applicable, its application actually modifies the decomposition.

Now let $m_{\mathcal{D}} = \mathsf{mln}^2 \cdot |\mathbf{L}| + \mathsf{mln} \cdot 3 \cdot |\mathbf{L}|$. The decomposition construction algorithm can modify the decomposition at most $m_{\mathcal{D}}$ times: it can add each of the $\mathsf{mln}^2 \cdot |\mathbf{L}|$ possible edges, and it can add $|\mathbf{L}|$ concepts to $\mathsf{core}(v)$, $\mathsf{knw}(v)$, and $\mathsf{poss}(v)$ for each context $v \in \mathcal{V}$. Thus, after at most $m_{\mathcal{D}}$ modifications, no rule can further modify the decomposition. Therefore, since each rule application modifies the decomposition, the decomposition construction algorithm terminates after at most $m_{\mathcal{D}}$ rule applications. Furthermore, due to the fact that each rule precondition refers to a bounded number of vertices, the preconditions of each rule instance can be checked in polynomial time. Since $\mathsf{strategy}$ is computable in polynomial time, each rule can be applied in polynomial time as well.

Thus, the algorithm requires a polynomial number of modifications of the decomposition, each of which requires polynomial time. Since $|\mathbf{L}| < \|\mathcal{O}\| + \|\mathcal{Q}\|$, the algorithm can be implemented so that it runs in time polynomial in mln and $\|\mathcal{O}\| + \|\mathcal{Q}\|$. □

Claim 31. *Decomposition* $\mathcal{D}$ *is unique.*

Proof. In all rules apart from **S**, the negative precondition merely checks whether the rule's postcondition is satisfied. Therefore, the rules are monotonic and they have a unique least fixpoint—that is, the order of rule applications is irrelevant. Furthermore, rule **S** is applied only if no other rule is applicable, and $\mathsf{precedence}$ ensures that the rule is applied in a deterministic way. Thus, the result of the decomposition construction algorithm is uniquely determined by $\mathcal{O}$, $\mathcal{Q}$, and $\mathcal{C}$. □

Claim 32. *Decomposition* $\mathcal{D}$ *is sound.*

Proof. The proof is by induction on rule application. After the initial step 1, we have $\mathcal{E} = \emptyset$ and $\mathsf{core}(v) = \mathsf{knw}(v)$ for each context v, so $\mathcal{D}$ is sound. We next consider all rules from Table 4.

($\mathbf{H}_k$) If $K \subseteq \mathsf{knw}(v)$ and $K \sqsubseteq L \in \mathcal{O}$, then, by the induction hypothesis, we have $\mathcal{O} \models \mathsf{core}(v) \sqsubseteq K$; but then, we clearly have $\mathcal{O} \models \mathsf{core}(v) \sqsubseteq L$, so adding L to $\mathsf{knw}(v)$ preserves soundness.

($\mathbf{P}_k$) If $\langle v, u, \exists R.A \rangle \in \mathcal{E}$, $\exists R.A \in \mathsf{knw}(v)$, and $\forall \mathsf{inv}(R).C \in \mathsf{knw}(u)$, then, by the induction hypothesis, all of the following entailments hold:

$$\mathcal{O} \models \mathsf{core}(v) \sqcap \exists R.A \sqsubseteq \exists R.[\mathsf{core}(u) \sqcap A] \qquad \mathcal{O} \models \mathsf{core}(v) \sqsubseteq \exists R.A \qquad \mathcal{O} \models \mathsf{core}(u) \sqsubseteq \forall \mathsf{inv}(R).C$$

These entailments imply $\mathcal{O} \models \mathsf{core}(v) \sqsubseteq \exists R.\forall \mathsf{inv}(R).C$, which then implies $\mathcal{O} \models \mathsf{core}(v) \sqsubseteq C$. Consequently, adding C to $\mathsf{knw}(v)$ preserves soundness.

(**S**) By the induction hypothesis, we have $\mathcal{O} \models \mathsf{core}(v) \sqsubseteq \forall R.B$ for each $B \in \mathbf{B}_k$, so $\mathcal{O} \models \mathsf{core}(v) \sqcap \exists R.A \sqsubseteq \exists R.[A \sqcap \mathbf{B}_k]$ holds; but then, $\mathcal{O} \models \mathsf{core}(v) \sqcap \exists R.A \sqsubseteq \exists R.[\mathsf{core}(u) \sqcap A]$ holds since we have $\mathsf{core}(u) = \mathsf{core}' \subseteq \{A\} \cup \mathbf{B}_k$ by the requirement on $\mathsf{strategy}$ in Definition 27. Consequently, adding the edge $\langle v, u, \exists R.A \rangle$ to $\mathcal{E}$ preserves soundness.

Since no other rule affects $\mathcal{E}$ or knw, this concludes the proof of this claim. □

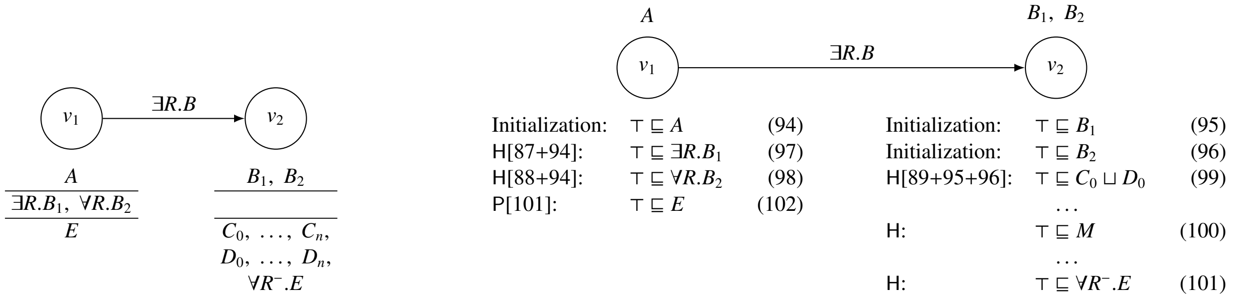
Claim 33. *Decomposition* $\mathcal{D}$ *is admissible.*

Proof. We check that $\mathcal{D}$ satisfies all conditions of Definition 20. The ontology condition and the structural conditions (S2) and (S3) hold trivially since the construction is closed under rules $\mathbf{H}_p$, $\mathbf{P}_p$, and **S**, respectively.

For condition (S1), we show that $\mathsf{core}(v) \subseteq \mathsf{knw}(v)$ and $\mathsf{knw}(v) \subseteq \mathsf{poss}(v)$ hold for each context $v \in \mathcal{V}$. The first inclusion holds because each set $\mathsf{knw}(v)$ is initialized in step 1 and in rule **S** with $\mathsf{knw}(v) := \mathsf{core}(v)$, and $\mathsf{core}(v)$ remains constant during the construction while $\mathsf{knw}(v)$ only increases. The second inclusion holds because each set $\mathsf{poss}(v)$ is initialized with $\mathsf{poss}(v) := \mathsf{knw}(v)$, and the rules $\mathbf{H}_p$ and $\mathbf{P}_p$ extending $\mathsf{poss}(v)$ are applicable whenever the rules $\mathbf{H}_k$ and $\mathbf{P}_k$ extending $\mathsf{knw}(v)$ are applicable.

The initial context structure produced by $\mathsf{initialization}$ contains no edges, and whenever an edge $\langle v, u, \exists R.A \rangle$ is added to $\mathcal{E}$ in rule **S**, $\mathsf{strategy}$ ensures that $\prec_u$ is R-admissible. Thus, the ordering condition holds throughout the construction.

Finally, for the covering condition, consider an arbitrary query $K \sqsubseteq M \in \mathcal{Q}$. After step 1, context $v = \vartheta(K \sqsubseteq M)$ covers $K \sqsubseteq M$ in the sense of Definition 7—that is, $\mathsf{core}(v) \subseteq K$ and M is $\prec_v$-minimal. Then, step 2 ensures also that $K \subseteq \mathsf{poss}(v)$; thus, v covers $K \sqsubseteq M$, as required by Definition 20. □



v_1
$\exists R.B$
v_2
A
$\exists R.B_1,\ \forall R.B_2$
E
$B_1,\ B_2$
$C_0,\ \ldots,\ C_n,$
$D_0,\ \ldots,\ D_n,$
$\forall R^{-}.E$
A
$B_1,\ B_2$
v_1
$\exists R.B$
v_2
Initialization: $\top \sqsubseteq A$ (94)
H[87+94]: $\top \sqsubseteq \exists R.B_1$ (97)
H[88+94]: $\top \sqsubseteq \forall R.B_2$ (98)
P[101]: $\top \sqsubseteq E$ (102)
Initialization: $\top \sqsubseteq B_1$ (95)
Initialization: $\top \sqsubseteq B_2$ (96)
H[89+95+96]: $\top \sqsubseteq C_0 \sqcup D_0$ (99)
...
H: $\top \sqsubseteq M$ (100)
...
H: $\top \sqsubseteq \forall R^{-}.E$ (101)

(a) Decomposition $\mathcal{D}_1$

(b) Inferences

Figure 6: An ϵ-Free Decomposition of $\mathcal{O}$ and q and the Related Inferences

We are now ready to combine all of the results presented thus far and formulate what we believe to be the first result on fixed-parameter tractability of subsumption reasoning in description logics.

Theorem 34. *For each control $\mathcal{C}$, the following problem is fixed-parameter tractable:*

- Inputs: *a normalized $\mathcal{ALCI}$ ontology $\mathcal{O}$ and a set of queries $\mathcal{Q}$*
- Parameter: *an integer* mwd
- Problem: *return "yes" if the $\mathcal{C}$-decomposition of $\mathcal{O}$ and $\mathcal{Q}$ has width at most* mwd*, and if also $\mathcal{O} \models K \sqsubseteq M$ holds for each query $K \sqsubseteq M \in \mathcal{Q}$*

Proof. Immediate by Theorem 29, and Propositions 24 and 25. □

Please note that Theorem 34 requires the input ontology to be normalized. As we mentioned in Section 2.1, many different normalizing transformations have been proposed in the literature, and they can produce syntactically very different ontologies. Thus, for an ontology that is not normalized, the width, the length, and our fixed-parameter tractability results implicitly depend on the normalization transformation. Investigating how normalization affects width and length, as well as what kind of transformation can minimize these two parameters is interesting, but we leave it for our future work.

5. Analyzing Or-Branching Using General Decompositions

We now generalize our framework to also allow for a quantitative analysis of or-branching. We first discuss the intuitions behind our ideas, and then we proceed with a formal presentation of our results.

5.1. Intuitions

Example 35. *Let $\mathcal{O}$ be the ontology consisting of axioms* (87)–(93)*, and let $q = A \sqsubseteq E$. One can readily check that $\mathcal{O} \models q$ holds.*

$$A \sqsubseteq \exists R.B_1 \quad (87) \qquad B_1 \sqcap B_2 \sqsubseteq C_0 \sqcup D_0 \quad (89) \qquad C_n \sqsubseteq \forall R^{-}.E \quad (92)$$

$$A \sqsubseteq \forall R.B_2 \quad (88) \qquad C_{i-1} \sqsubseteq C_i \sqcup D_i \quad \textit{for } 1 \leq i \leq n \quad (90) \qquad D_n \sqsubseteq \forall R^{-}.E \quad (93)$$

$$D_{i-1} \sqsubseteq C_i \sqcup D_i \quad \textit{for } 1 \leq i \leq n \quad (91)$$

The $\mathcal{C}$-decomposition $\mathcal{D}_1$ of $\mathcal{O}$ and q obtained using the eager strategy is shown in Figure 6a, and the inferences of Algorithm 21 on $\mathcal{D}_1$ are shown in Figure 6b. Clause (100) stands for clauses with $M \subseteq \{C_0, \ldots, C_n, D_0, \ldots, D_n, \forall R^{-}.E\}$; exactly which clauses of this form are derived depends on the literal ordering $\prec_{v_2}$. For example, with $\forall R^{-}.E \prec_{v_2} C_n \prec_{v_2} D_n \prec_{v_2} \ldots \prec_{v_2} C_0 \prec_{v_2} D_0$ we derive a linear number of clauses, and with $\forall R^{-}.E \prec_{v_2} C_0 \prec_{v_2} D_0 \prec_{v_2} \ldots \prec_{v_2} C_n \prec_{v_2} D_n$ we derive an exponential number of clauses. In practice, it is difficult to identify a literal ordering that minimizes the number of derived clauses, so the only guarantee on the algorithm's performance is exponential in n; this is in agreement with the fact that $\mathsf{wd}(\mathcal{D}_1) = 2n + 3$.

Clauses in $\mathcal{S}(v_2)$ essentially solve a propositional problem consisting of clauses (89)–(93) and clause $\alpha = B_1 \sqcap B_2 \sqsubseteq \forall R^{-}.E$: the antecedent of α captures the information that can be propagated from v_1 to v_2, and the consequent of α captures the information that can be propagated from v_2 to v_1. As we explained in Section 2.4, the treewidth of propositional problems serves as a measure of the difficulty of such problems, and the bottom-up algorithm provides an FPT decision procedure for propositional satisfiability. Thus, to explain the difficulty of or-branching in description logics, we next incorporate tree decompositions into our framework. To this end, we replace context v_2 with a tree decomposition of the mentioned propositional problem, where we label the newly introduced edges by a special symbol ϵ; similarly to edges in a tree decomposition, our ϵ-edges will always be symmetric. Decomposition $\mathcal{D}_2$ obtained in this way is shown in Figure 7a. The set of contexts $\{v_2^0, \dots, v_2^n\}$ obtained by replacing v_2 is called an *ϵ-component* of $\mathcal{D}_2$. Intuitively, each ϵ-component $\mathcal{W}$ of $\mathcal{D}_2$ corresponds to one or more domain elements in a model of $\mathcal{O}$, and each context $v \in \mathcal{W}$ provides a partial interpretation for the literals in $\mathsf{poss}(v)$. Condition (N2) of Definition 19 requires all contexts in an ϵ-component $\mathcal{W}$ of a sound decomposition to have the same core; thus, all partial descriptions of the elements represented by $\mathcal{W}$ must coincide on the "assumed" literals. However, the contexts in $\mathcal{W}$ can differ on known and possible literals, which will allow us to reduce decomposition width.

We will extend the notion of decomposition admissibility so that it guarantees completeness of a modified consequence-based algorithm shown in Table 5 on page 28. The Hyper and Pred rules are as in Table 3, but modified so that a clause is added to some $\mathcal{S}(v)$ only if all of its literals are possible at context v. In addition, we introduce a new Epsilon rule, which essentially performs hyperresolution across ϵ-edges: given an edge $\langle v, u, \epsilon \rangle$, a clause in $\bigsqcap_{i=1}^n L_i \sqsubseteq M \in \mathcal{S}(u)$ is chosen such that all literals in M are possible at context v, each literal L_i is resolved away using a clause in $\mathcal{S}(v)$, and the result is added to $\mathcal{S}(v)$. The Epsilon rule can be seen as a practical variant of the bottom-up algorithm for propositional satisfiability described in Section 2.4.

The inferences of our new algorithm on $\mathcal{O}$, q, and $\mathcal{D}_2$ are shown in Figure 7b. As before, we first initialize each $\mathcal{S}(v)$ with $\top \sqsubseteq L$ for each literal L known at context v, thus obtaining clauses (103)–(113). Moreover, for each edge $\langle v, u, \epsilon \rangle$ and each literal L that is possible at both u and v, we initialize $\mathcal{S}(u)$ and $\mathcal{S}(v)$ with $L \sqsubseteq L$, thus obtaining clauses (114)–(125). Finally, using rules from Table 5 on page 28, we derive clauses (126)–(138). At each context v_2^i with $0 \leq i < n$, inferences follow the same pattern: using clauses (90) and (91) from $\mathcal{O}$, we derive a clause of the form $\top \sqsubseteq C_{i+1} \sqcup D_{i+1}$, which the Epsilon rule then "transfers" over an ϵ-edge into context v_2^{i+1}. Finally, in context v_2^n we derive clause (137), and then by the Pred rule we derive the goal clause (138). In order to ensure that the Epsilon rule derives all relevant consequences, our notion of decomposition admissibility imposes an additional restriction on the literal orderings of contexts. For example, clause (128) is relevant for the Epsilon rule and is derived using the Hyper rule from clause (127); however, to facilitate the latter derivation, concepts D_1 and C_1 must not be larger in $\prec_{v_2^0}$ than concept D_0. Thus, the new restriction on literal orderings is analogous to R-admissibility for $\exists R.A$-edges, but adapted to ϵ-edges. To estimate the complexity of our algorithm, we note that $\mathsf{wd}(\mathcal{D}_2) = 4$; hence, the number of clauses derived at each context is bounded by $4^{\mathsf{wd}(\mathcal{D}_2)} = 4^4$, and this bound does not depend on the literal ordering or the number n. In this way, $\mathsf{wd}(\mathcal{D}_2)$ provides us with a better estimate of the or-branching involved in deciding $\mathcal{O} \models q$.

As in Section 4, the definitions of soundness and admissibility for general decompositions do not directly provide us with an algorithm for computing decompositions. As a possible remedy, in Section 5.3 we introduce the notion of an *ϵ-refinement*, which is obtained from an ϵ-free decomposition by replacing each context v with a tree decomposition of the propositional problem associated with v. In our example, $\mathcal{D}_2$ is an ϵ-refinement of $\mathcal{D}_1$. We will show that computing an ϵ-refinement involves determining a linear number of tree decompositions and is thus fixed-parameter tractable w.r.t. decomposition width.

It is again instructive to consider then case when $\mathcal{O}$ and $\mathcal{Q}$ contain only propositional clauses. Then, for each admissible decomposition $\mathcal{D}$ of $\mathcal{O}$ and $\mathcal{Q}$, the $\exists R.A$-edges are not needed and can be removed, but $\mathcal{D}$ can contain several ϵ-components, each of which corresponds to a tree decomposition of a subset of $\mathcal{O}$ but with the known literals removed. Since the known literals are removed, the width of such $\mathcal{D}$ can be lower than the treewidth of $\mathcal{O}$, and so our decompositions generalize tree decompositions of propositional problems. Furthermore, only the Hyper and Epsilon are used when our consequence-based calculus is applied to such $\mathcal{D}$, and the calculus can be seen as a practical variant of the algorithm for propositional logic presented in Section 2.4.

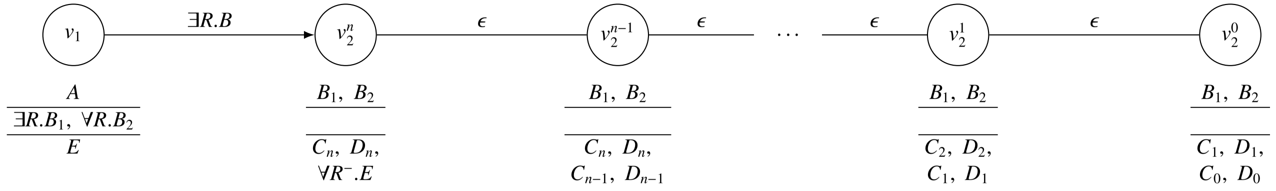
5.2. Formalization

Decompositions and decomposition soundness (with or without ϵ-edges) were already introduced in Section 4.2, so we next focus on admissibility of decompositions with ϵ-edges. In the rest of this section we fix a normalized $\mathcal{ALCI}$ ontology $\mathcal{O}$ and a finite set of queries $\mathcal{Q}$; furthermore, we let $\mathbf{L}$ be the set of literals that occur in $\mathcal{O} \cup \mathcal{Q}$.

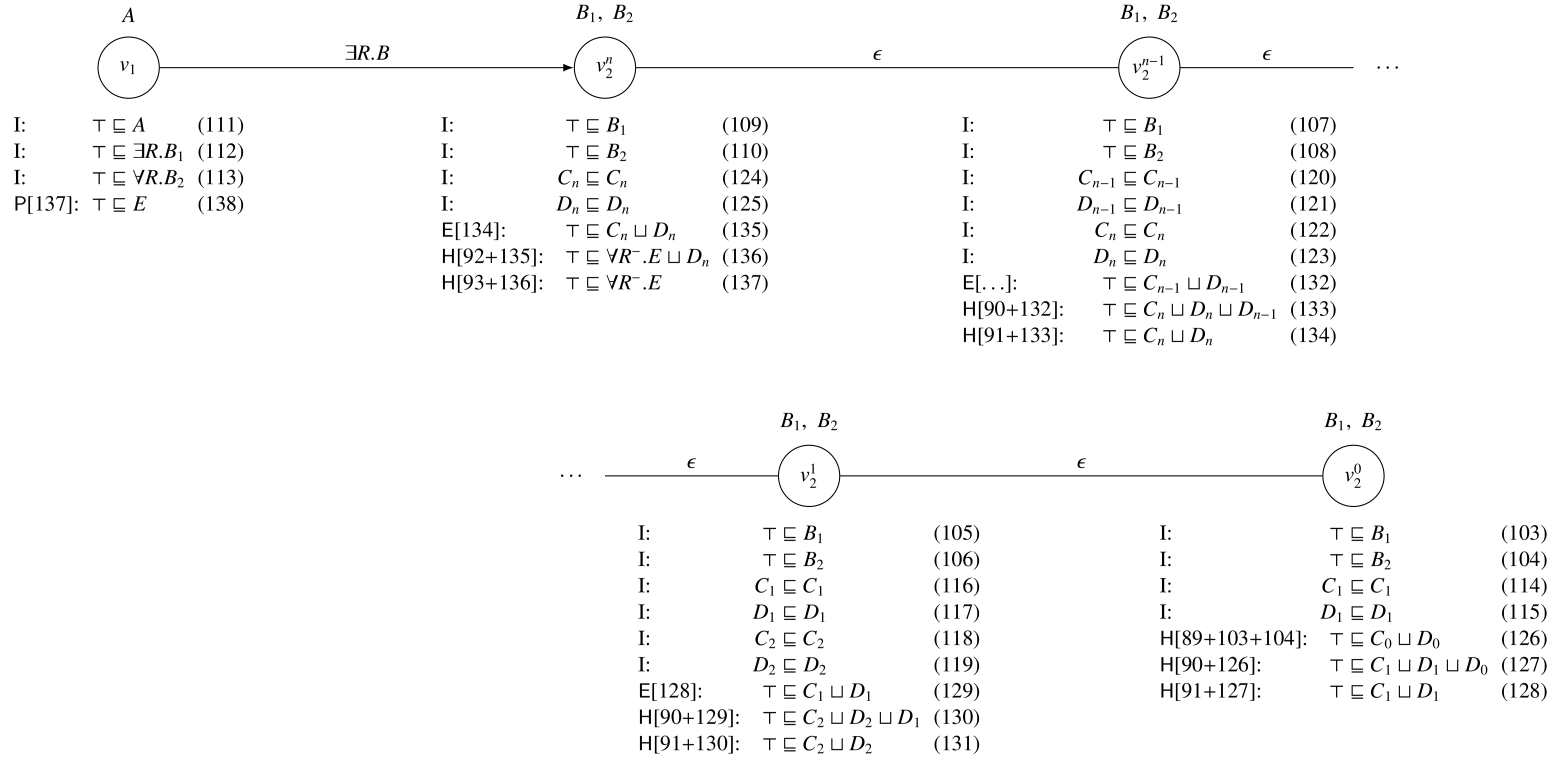
Definition 36. *Let $\mathcal{D} = \langle \mathcal{V}, \mathcal{E}, \mathsf{core}, \mathsf{knw}, \mathsf{poss}, \prec, \vartheta \rangle$ be a decomposition of $\mathcal{O}$ and $\mathcal{Q}$, and let $\mathcal{W}$ be an arbitrary set such that $\mathcal{W} \subseteq \mathcal{V}$. Then, let $\mathsf{poss}(\mathcal{W}) := \bigcup_{w \in \mathcal{W}} \mathsf{poss}(w)$, and let $\mathsf{knw}(\mathcal{W})$ and $\mathsf{core}(\mathcal{W})$ be defined analogously. The* ϵ-projection *of $\mathcal{D}$ w.r.t. $\mathcal{W}$, written $\mathcal{D}_{\mathcal{W}}$, is the graph whose set of vertices is $\mathcal{W}$ and that contains the undirected edge $\{u, v\}$ for each $\langle u, v, \epsilon \rangle \in \mathcal{E}$ with $u, v \in \mathcal{W}$. Set $\mathcal{W}$ is* ϵ-connected *if $\mathcal{D}_{\mathcal{W}}$ is connected; furthermore, $\mathcal{W}$ is an* ϵ-component *of $\mathcal{D}$ if $\mathcal{W}$ is ϵ-connected, and no ϵ-connected set of vertices $\mathcal{W}'$ exists such that $\mathcal{W} \subsetneq \mathcal{W}' \subseteq \mathcal{V}$.*

Let $v \in \mathcal{V}$ be an arbitrary context of $\mathcal{D}$, let $\mathcal{W}$ be the ϵ-component of $\mathcal{D}$ such that $v \in \mathcal{W}$, and let $K \sqsubseteq M$ be an arbitrary query; then, v is sound *for $K \sqsubseteq M$ if $\mathsf{core}(v) \subseteq K$; v is* complete *for $K \sqsubseteq M$ if $K \subseteq \mathsf{poss}(v)$, $M \cap \mathsf{poss}(\mathcal{W}) \subseteq \mathsf{poss}(v)$, and M is $\prec_v$-minimal; and v* covers *$K \sqsubseteq M$ if v is both sound and complete for $K \sqsubseteq M$. Finally, $\mathcal{D}$ is* admissible *if all of the following conditions are satisfied.*

- *Epsilon conditions:*
  - *(E1) For each $\langle w, u, \epsilon \rangle \in \mathcal{E}$, we have $\langle u, w, \epsilon \rangle \in \mathcal{E}$.*
  - *(E2) For each ϵ-component $\mathcal{W}$ of $\mathcal{D}$, graph $\mathcal{D}_{\mathcal{W}}$ is an undirected tree.*
  - *(E3) For each ϵ-component $\mathcal{W}$ of $\mathcal{D}$ and each literal $L \in \mathsf{poss}(\mathcal{W})$, set $\{w \in \mathcal{W} \mid L \in \mathsf{poss}(w)\}$ is ϵ-connected.*
- *Structural conditions:*



(a) Decomposition $\mathcal{D}_2$



I: $\top \sqsubseteq A$ (111)
I: $\top \sqsubseteq \exists R.B_1$ (112)
I: $\top \sqsubseteq \forall R.B_2$ (113)
P[137]: $\top \sqsubseteq E$ (138)

I: $\top \sqsubseteq B_1$ (109)
I: $\top \sqsubseteq B_2$ (110)
I: $C_n \sqsubseteq C_n$ (124)
I: $D_n \sqsubseteq D_n$ (125)
E[134]: $\top \sqsubseteq C_n \sqcup D_n$ (135)
H[92+135]: $\top \sqsubseteq \forall R^{-}.E \sqcup D_n$ (136)
H[93+136]: $\top \sqsubseteq \forall R^{-}.E$ (137)

I: $\top \sqsubseteq B_1$ (107)
I: $\top \sqsubseteq B_2$ (108)
I: $C_{n-1} \sqsubseteq C_{n-1}$ (120)
I: $D_{n-1} \sqsubseteq D_{n-1}$ (121)
I: $C_n \sqsubseteq C_n$ (122)
I: $D_n \sqsubseteq D_n$ (123)
E[...]: $\top \sqsubseteq C_{n-1} \sqcup D_{n-1}$ (132)
H[90+132]: $\top \sqsubseteq C_n \sqcup D_n \sqcup D_{n-1}$ (133)
H[91+133]: $\top \sqsubseteq C_n \sqcup D_n$ (134)

I: $\top \sqsubseteq B_1$ (105)
I: $\top \sqsubseteq B_2$ (106)
I: $C_1 \sqsubseteq C_1$ (116)
I: $D_1 \sqsubseteq D_1$ (117)
I: $C_2 \sqsubseteq C_2$ (118)
I: $D_2 \sqsubseteq D_2$ (119)
E[128]: $\top \sqsubseteq C_1 \sqcup D_1$ (129)
H[90+129]: $\top \sqsubseteq C_2 \sqcup D_2 \sqcup D_1$ (130)
H[91+130]: $\top \sqsubseteq C_2 \sqcup D_2$ (131)

I: $\top \sqsubseteq B_1$ (103)
I: $\top \sqsubseteq B_2$ (104)
I: $C_1 \sqsubseteq C_1$ (114)
I: $D_1 \sqsubseteq D_1$ (115)
H[89+103+104]: $\top \sqsubseteq C_0 \sqcup D_0$ (126)
H[90+126]: $\top \sqsubseteq C_1 \sqcup D_1 \sqcup D_0$ (127)
H[91+127]: $\top \sqsubseteq C_1 \sqcup D_1$ (128)

(b) Inferences

Figure 7: A Decomposition of $\mathcal{O}$ and q with ϵ-Edges and the Related Inferences

Table 5: Consequence-based rules for decompositions with ϵ-edges

Hyper	If	$\bigsqcap_{i=1}^{n} A_i \sqsubseteq M \in \mathcal{O}$ with $\underline{M \subseteq \mathsf{poss}(v)}$,
		$K_i \sqsubseteq M_i \sqcup A_i \in \mathcal{S}(v)$ with $A_i \not\prec_v M_i$ for $1 \leq i \leq n$,
		and $\bigsqcap_{i=1}^{n} K_i \sqsubseteq M \sqcup \bigsqcup_{i=1}^{n} M_i \mathrel{\hat{\notin}} \mathcal{S}(v)$,
	then	add $\bigsqcap_{i=1}^{n} K_i \sqsubseteq M \sqcup \bigsqcup_{i=1}^{n} M_i$ to $\mathcal{S}(v)$.
Pred	If	$\langle v, u, \exists R.A \rangle \in \mathcal{E}$,
		$A \sqcap \bigsqcap_{i=1}^{n} B_i \sqsubseteq \bigsqcup_{j=1}^{m} \forall \mathsf{inv}(R).C_j \in \mathcal{S}(u)$ or $\bigsqcap_{i=1}^{n} B_i \sqsubseteq \bigsqcup_{j=1}^{m} \forall \mathsf{inv}(R).C_j \in \mathcal{S}(u)$
		with $\underline{C_j \in \mathsf{poss}(v) \text{ for each } 1 \leq j \leq m}$,
		$K_0 \sqsubseteq M_0 \sqcup \exists R.A \in \mathcal{S}(v)$ with $\exists R.A \not\prec_v M_0$,
		$K_i \sqsubseteq M_i \sqcup \forall R.B_i \in \mathcal{S}(v)$ with $\forall R.B_i \not\prec_v M_i$ for $1 \leq i \leq n$,
		and $\bigsqcap_{i=0}^{n} K_i \sqsubseteq \bigsqcup_{i=0}^{n} M_i \sqcup \bigsqcup_{j=1}^{m} C_j \mathrel{\hat{\notin}} \mathcal{S}(v)$,
	then	add $\bigsqcap_{i=0}^{n} K_i \sqsubseteq \bigsqcup_{i=0}^{n} M_i \sqcup \bigsqcup_{j=1}^{m} C_j$ to $\mathcal{S}(v)$.
Epsilon	If	$\langle v, u, \epsilon \rangle \in \mathcal{E}$,
		$\bigsqcap_{i=1}^{n} L_i \sqsubseteq M \in \mathcal{S}(u)$ with $\underline{M \subseteq \mathsf{poss}(v)}$,
		$K_i \sqsubseteq M_i \sqcup L_i \in \mathcal{S}(v)$ with $L_i \not\prec_v M_i$ for $1 \leq i \leq n$,
		and $\bigsqcap_{i=1}^{n} K_i \sqsubseteq M \sqcup \bigsqcup_{i=1}^{n} M_i \mathrel{\hat{\notin}} \mathcal{S}(v)$,
	then	add $\bigsqcap_{i=1}^{n} K_i \sqsubseteq M \sqcup \bigsqcup_{i=1}^{n} M_i$ to $\mathcal{S}(v)$.

(S1) For each context $v \in \mathcal{V}$, we have $\mathsf{core}(v) \subseteq \mathsf{knw}(v) \subseteq \mathsf{poss}(v)$.

(S2) For each ϵ-component $\mathcal{U}$ of $\mathcal{D}$, each edge $\langle w, u, \exists R.A \rangle \in \mathcal{E}$ with $u \in \mathcal{U}$, and each concept $\forall \mathsf{inv}(R).C \in \mathsf{poss}(\mathcal{U})$*, we have* $C \in \mathsf{poss}(w)$ *and* $\forall \mathsf{inv}(R).C \in \mathsf{poss}(u)$.

(S3) For each ϵ-component $\mathcal{W}$ of $\mathcal{D}$ and each $\exists R.A \in \mathsf{poss}(\mathcal{W})$*, some edge $\langle w, u, \exists R.A \rangle \in \mathcal{E}$ with $w \in \mathcal{W}$ exists such that*

- $\mathsf{poss}(w) \supseteq \{\exists R.A\} \cup \{\forall R.B \mid \forall R.B \in \mathsf{poss}(\mathcal{W})\}$*, and*
- $\mathsf{poss}(u) \supseteq \{A\} \cup \{B \mid \forall R.B \in \mathsf{poss}(\mathcal{W})\}$.

- *Ordering conditions:*

  (P1) For each $\exists R.A$-edge $\langle w, u, \exists R.A \rangle \in \mathcal{E}$, literal ordering $\prec_u$ is R-admissible.

  (P2) For each ϵ-edge $\langle w, u, \epsilon \rangle \in \mathcal{E}$, set $\mathsf{poss}(w) \cap \mathsf{poss}(u)$ *is $\prec_u$-minimal.*

- *Ontology condition: For each ϵ-component $\mathcal{W}$ of $\mathcal{D}$ and each clause $K \sqsubseteq M \in \mathcal{O}$ with* $K \subseteq \mathsf{poss}(\mathcal{W})$*, a context $w \in \mathcal{W}$ exists such that* $K \cup M \subseteq \mathsf{poss}(w)$.

- *Covering condition: Each query $q \in \mathcal{Q}$ is covered in the context $\vartheta(q)$.*

Definition 36 can be intuitively understood as follows. The set of contexts in $\mathcal{D}$ can be partitioned into ϵ-components—maximal sets of contexts connected by ϵ-edges. Each ϵ-component can be understood as a tree decomposition of a propositional problem, so conditions (E1)–(E3) require the contexts in the ϵ-component to form an undirected tree satisfying the connectedness condition (T2) of tree decompositions. Structural conditions (S1)–(S3) extend the respective conditions from Definition 20. To understand condition (S2), let $\langle w, u, \exists R.A \rangle$ be an arbitrary edge of $\mathcal{D}$, let $\mathcal{U}$ be the ϵ-component of $\mathcal{D}$ that u belongs to, and assume that $\forall \mathsf{inv}(R).C$ is possible at some context $u' \in \mathcal{U}$ different from u. Since u' and u describe the same elements in a model of $\mathcal{O}$, the elements represented by context v might need to satisfy C, which is taken care of by the Pred rule. To ensure that the rule can be applied to edge $\langle w, u, \exists R.A \rangle$, condition (S2) requires $\forall \mathsf{inv}(R).C$ to be possible at u, and C to be possible at v (just like in Definition 20). To understand condition (S3), let $\mathcal{W}$ be an arbitrary ϵ-component of $\mathcal{D}$, assume that $\exists R.A$ is possible at some context $w' \in \mathcal{W}$, and assume that $\mathcal{D}$ contains an edge $\langle w, u, \exists R.A \rangle$ with w different from w'. Contexts w and w' jointly describe elements of a model of $\mathcal{O}$, so the existential restriction $\exists R.A$ also “applies” to w and the mentioned edge as well; thus, all literals $\forall R.B$ possible at $\mathcal{W}$ must be possible at w, and all corresponding concepts B must be possible at u in order to satisfy the semantics of $\exists R.A$-edges. The ontology condition is modified in similar vein; in particular, assume that $\mathcal{D}$ contains an ϵ-component $\mathcal{W}$ such that each antecedent literal of a clause $K \sqsubseteq M \in \mathcal{O}$ is possible at some context in $\mathcal{W}$; then, the domain elements represented by $\mathcal{W}$ may satisfy K; but then, we must find a context $w \in \mathcal{W}$ in which we can apply the Hyper rule to $K \sqsubseteq M$. The notion of covering is extended analogously. Finally, the ordering condition restricts the literal ordering as we discussed in Section 5.1. Please note that the notions of admissibility introduced in Definitions 20 and 36 coincide on ϵ-free decompositions.

To generalize our consequence-based algorithm as described in Section 5.1, we modify the Hyper and the Pred rules as shown in Table 5 (the modifications are underlined), and we add a new Epsilon rule. Theorem 37 provides us with the completeness claim for the new algorithm; the proof is technically involved, so we defer it to Appendix B. In addition, Proposition 38 provides us with a matching soundness claim.

Theorem 37 (Completeness)**.** *Let $\mathcal{D} = \langle \mathcal{V}, \mathcal{E}, \mathsf{core}, \mathsf{knw}, \mathsf{poss}, \prec, \vartheta \rangle$ be an admissible decomposition of $\mathcal{O}$ and $\mathcal{Q}$, and let $\mathcal{S}$ be a clause system for $\mathcal{D}$ such that*

(I1) no inference rule in Table 5 is applicable to $\mathcal{S}$,

(I2) $\top \sqsubseteq L \mathrel{\hat{\in}} \mathcal{S}(v)$ for each context $v \in \mathcal{V}$ and each literal $L \in \mathsf{knw}(v)$,

(I3) $B \sqsubseteq B \mathrel{\hat{\in}} \mathcal{S}(u)$ for each edge $\langle v, u, \exists R.A \rangle \in \mathcal{E}$ and each concept $B \in \mathsf{poss}(u)$ such that either $B = A$ or $\forall R.B \in \mathsf{poss}(v)$, and

(I4) $L \sqsubseteq L \mathrel{\hat{\in}} \mathcal{S}(u)$ for each edge $\langle v, u, \epsilon \rangle \in \mathcal{E}$ and each literal $L \in \mathsf{poss}(v) \cap \mathsf{poss}(u)$.

Then, $K \sqsubseteq M \mathrel{\hat{\in}} \mathcal{S}(v)$ holds for each query $K \sqsubseteq M$ and each context $v \in \mathcal{V}$ that satisfy all of the following three conditions:

- *$\mathcal{O} \models K \sqsubseteq M$,*
- *context v is complete for query $K \sqsubseteq M$, and*
- *$K \sqsubseteq L \mathrel{\hat{\in}} \mathcal{S}(v)$ for each literal $L \in K$.*

Proposition 38 (Soundness)**.** *Let $\mathcal{D} = \langle \mathcal{V}, \mathcal{E}, \mathsf{core}, \prec \rangle$ be a sound decomposition of $\mathcal{O}$ and $\mathcal{Q}$, and let $\mathcal{S}_1$ be a clause system for $\mathcal{D}$ sound for $\mathcal{O}$. Then, each clause system $\mathcal{S}_2$ obtained by applying an inference rule from Table 5 to $\mathcal{D}$ and $\mathcal{S}_1$ is sound for $\mathcal{O}$.*

Proof. Rules Hyper and Pred in Table 5 are more restrictive than the corresponding rules in Table 3, so the proof of Proposition 9 applies. Assume that the Epsilon rule is applied as shown in Table 5, so we show that $\mathcal{O} \models \mathsf{core}(v) \sqcap \bigsqcap_{i=1}^n K_i \sqsubseteq M \sqcup \bigsqcup_{i=1}^n M_i$. To this end, let $\mathcal{I} = \langle \Delta^\mathcal{I}, \cdot^\mathcal{I} \rangle$ be an arbitrary model of $\mathcal{O}$ and consider an arbitrary element $\delta \in \Delta^\mathcal{I}$ such that $\delta \in [\mathsf{core}(v) \sqcap \bigsqcap_{i=1}^n K_i]^\mathcal{I}$. Now $\mathcal{S}_1$ is sound for $\mathcal{O}$, so, for each $1 \leq i \leq n$, we have $\mathcal{I} \models \mathsf{core}(v) \sqcap K_i \sqsubseteq M_i \sqcup L_i$, which together with our assumption implies $\delta \in (M_i \sqcup L_i)^\mathcal{I}$. If $\delta \in M_i^\mathcal{I}$ for some $1 \leq i \leq n$, then $\delta \in (\bigsqcup_{i=1}^n M_i)^\mathcal{I}$. Otherwise, we have $\delta \in L_i^\mathcal{I}$ for each $1 \leq i \leq n$; but then, $\mathcal{I} \models \mathsf{core}(u) \sqcap \bigsqcap_{i=1}^n L_i \sqsubseteq M$ holds by the induction assumption and the fact that $\mathcal{S}_1$ is sound for $\mathcal{O}$, and $\mathsf{core}(v) = \mathsf{core}(u)$ holds by the soundness condition (N2); together, these observations imply $\delta \in M^\mathcal{I}$. Either way, $\delta \in (M \sqcup \bigsqcup_{i=1}^n M_i)^\mathcal{I}$ holds. Since δ was chosen arbitrarily, we have $\mathcal{I} \models \mathsf{core}(v) \sqcap \bigsqcap_{i=1}^n K_i \sqsubseteq M \sqcup \bigsqcup_{i=1}^n M_i$, as required. □

We next extend the consequence-based algorithm for ϵ-free decompositions (i.e., Algorithm 21) so that it can be applied to decompositions with ϵ-edges. Apart from applying the modified rules from Table 5, the only other difference is in step 3, which is needed for the Epsilon rule. The Elim rule and redundancy elimination are the same as in Section 3.3.

Algorithm 39. *The* decomposition-based reasoning algorithm *for $\mathcal{ALCI}$ takes $\mathcal{O}$, $\mathcal{Q}$, and a sound and admissible decomposition $\mathcal{D}$ of $\mathcal{O}$ and $\mathcal{Q}$. The algorithm (nondeterministically) computes a clause system $\mathcal{S}$ for $\mathcal{D}$ as follows.*

1. *Set $\mathcal{S}(v) := \{\top \sqsubseteq L \mid L \in \mathsf{knw}(v)\}$ for each context $v \in \mathcal{V}$.*
2. *For each edge $\langle v, u, \exists R.A \rangle \in \mathcal{E}$ and each concept $B \in \mathsf{uknw}(u)$ such that $B = A$ or $\forall R.B \in \mathsf{poss}(v)$, add $B \sqsubseteq B$ to $\mathcal{S}(u)$.*
3. *For each ϵ-edge $\langle v, u, \epsilon \rangle \in \mathcal{E}$ and each literal $L \in \mathsf{poss}(v) \cap \mathsf{uknw}(u)$, add $L \sqsubseteq L$ to $\mathcal{S}(u)$.*
4. *For each query $K \sqsubseteq M \in \mathcal{Q}$, let $v := \vartheta(K \sqsubseteq M)$; then, for each literal $L \in K \setminus \mathsf{knw}(v)$, add $L \sqsubseteq L$ to $\mathcal{S}(v)$.*
5. *Exhaustively apply rules* Hyper, Pred, *and* Epsilon *from Table 5 and rule* Elim *from Table 3.*

Lemma 12 (showing that if $K \sqsubseteq M \mathrel{\hat{\in}} \mathcal{S}(v)$ holds at some point during algorithm's execution, then this also holds at all future points) clearly applies to Algorithm 39 as well: the Epsilon rule only adds clauses, and the Elim rule is the same as in Section 3.2. Furthermore, the proof of Lemma 22 (showing that each clause in $\mathcal{S}(v)$ is of the form $\top \sqsubseteq L$ with $L \in \mathsf{knw}(v)$ or of the form $K \sqsubseteq M$ with $K \cup M \subseteq \mathsf{uknw}(v)$) can be straightforwardly extended to Algorithm 39. This allows us to establish in Proposition 40 the soundness and completeness of our algorithm, and then determine in Proposition 41 the algorithm's complexity.

Proposition 40. *Let $\mathcal{S}$ be a clause system obtained by applying Algorithm 39 to $\mathcal{O}$, $\mathcal{Q}$, and $\mathcal{D}$. Then, for each query $q \in \mathcal{Q}$, we have $\mathcal{O} \models q$ if and only if $q \mathrel{\hat{\in}} \mathcal{S}(\vartheta(q))$.*

Proof. By Lemma 12, steps 2–4 ensure conditions (I2)–(I4) and step 5 ensures the required condition on the queries in Theorem 37; then, the claim of this proposition follows from Theorem 37 and Proposition 38 as in the proof of Proposition 24. □

Proposition 41 (Termination)**.** *Algorithm 39 terminates in time polynomial in $4^{\mathsf{wd}(\mathcal{D})^2}$, $\mathsf{ln}(\mathcal{D})$, and $\|\mathcal{O}\| + \|\mathcal{Q}\|$.*

Proof. Analogous to the proof of Proposition 25. □

5.3. Constructing Decompositions via ϵ-Refinement

We next formalize the notion of ϵ-refinement of an ϵ-free decomposition $\mathcal{D}$ that we described in Section 5.1.

Definition 42. *Let $\mathcal{D} = \langle \mathcal{V}, \mathcal{E}, \mathsf{core}, \mathsf{knw}, \mathsf{poss}, \prec, \vartheta \rangle$ be an ϵ-free decomposition of $\mathcal{O}$ and $\mathcal{Q}$. For each edge $\langle v, u, \exists R.A \rangle \in \mathcal{E}$, sets $\delta_{v,u}^{\exists R.A}$ and $\varrho_{v,u}^{\exists R.A}$ are defined as follows:*

$$\begin{aligned}
\delta_{v,u}^{\exists R.A} &:= \{\exists R.A\} \cup \{\forall R.B \mid \forall R.B \in \mathsf{poss}(v)\} \cup \{C \mid \forall \mathsf{inv}(R).C \in \mathsf{poss}(u)\} \\
\varrho_{v,u}^{\exists R.A} &:= \{A\} \cup \{B \mid \forall R.B \in \mathsf{poss}(v)\} \cup \{\forall \mathsf{inv}(R).C \mid \forall \mathsf{inv}(R).C \in \mathsf{poss}(u)\}
\end{aligned}$$

Function $\mathcal{H}$ for $\mathcal{D}$ maps each context $v \in \mathcal{V}$ to an $\mathbf{L}$-hypergraph $\mathcal{H}_v$ such that each $\mathcal{H}_v$ is the smallest set satisfying all of the following conditions:

(R1) $(K \cup M) \cap \mathsf{uknw}(v) \in \mathcal{H}_v$ *for each context $v \in \mathcal{V}$ and each clause $K \sqsubseteq M \in \mathcal{O}$ such that $K \cup M \subseteq \mathsf{poss}(v)$,*

(R2) $\delta_{v,u}^{\exists R.A} \cap \mathsf{uknw}(v) \in \mathcal{H}_v$ *and* $\varrho_{v,u}^{\exists R.A} \cap \mathsf{uknw}(u) \in \mathcal{H}_u$ *for each edge $\langle v, u, \exists R.A \rangle \in \mathcal{E}$, and*

(R3) $(K \cup M) \cap \mathsf{uknw}(v) \in \mathcal{H}_v$ *for each query $q = K \sqsubseteq M \in \mathcal{Q}$ and context $v = \vartheta(q)$.*

A decomposition $\mathcal{D}' = \langle \mathcal{V}', \mathcal{E}', \mathsf{core}', \mathsf{knw}', \mathsf{poss}', \prec', \vartheta' \rangle$ of $\mathcal{O}$ and $\mathcal{Q}$ is an ϵ-refinement *of $\mathcal{D}$ if there exists a function $\mathcal{T}$ mapping each context $v \in \mathcal{V}$ to a tree decomposition $\mathcal{T}_v = \langle \mathcal{V}_v, \mathcal{E}_v, \mathcal{L}_v \rangle$ of $\mathcal{H}_v$ such that $\mathcal{V}_v \cap \mathcal{V}_w = \emptyset$ and $\mathcal{V}_v \subseteq \mathbf{X}$ for all $v, w \in \mathcal{V}$ with $v \neq w$, and all of the following conditions are satisfied.*

(R4) $\mathcal{V}' = \bigcup_{v \in \mathcal{V}} \mathcal{V}_v$.

(R5) For each context $v \in \mathcal{V}$ and each vertex $w \in \mathcal{V}_v$, we have

$$\mathsf{core}'(w) = \mathsf{core}(v), \qquad \mathsf{knw}'(w) = \mathsf{knw}(v), \qquad \textit{and} \qquad \mathsf{poss}'(w) = \mathsf{knw}(v) \cup \mathcal{L}_v(w).$$

(R6) For all $v, u \in \mathcal{V}$, each $w \in \mathcal{V}_v$, and each $z \in \mathcal{V}_u$, we have $\langle w, z, \epsilon \rangle \in \mathcal{E}'$ if and only if $v = u$ and $\{w, z\} \in \mathcal{E}_v$.

(R7) For all $v, u \in \mathcal{V}$, each $w \in \mathcal{V}_v$, each $z \in \mathcal{V}_u$, and each literal $\exists R.A \in \mathbf{L}$, we have $\langle w, z, \exists R.A \rangle \in \mathcal{E}'$ if and only if

$$\langle v, u, \exists R.A \rangle \in \mathcal{E}, \qquad \delta_{v,u}^{\exists R.A} \cap \mathsf{uknw}(v) \subseteq \mathcal{L}_v(w), \qquad \textit{and} \qquad \varrho_{v,u}^{\exists R.A} \cap \mathsf{uknw}(u) \subseteq \mathcal{L}_u(z).$$

(R8) For each context $v \in \mathcal{V}$, each vertex $w \in \mathcal{V}_v$, and all literals $L_1, L_2 \in \mathbf{L}$, we have $L_1 \prec'_w L_2$ if and only if $L_1 \prec_v L_2$ and no ϵ-edge $\langle u, w, \epsilon \rangle \in \mathcal{E}'$ exists such that $L_2 \in \mathsf{poss}'(u) \cap \mathsf{poss}'(w)$.

(R9) For each query $q = K \sqsubseteq M \in \mathcal{Q}$, context $v = \vartheta(q)$, and context $w = \vartheta'(q)$, we have $w \in \mathcal{V}_v$ and $(K \cup M) \cap \mathsf{uknw}(v) \subseteq \mathcal{L}_v(w)$.

Definition 42 can be intuitively understood as follows. Each hypergraph $\mathcal{H}_v$ captures the structure of a propositional problem that must be solved at context v: condition (R1) ensures that, for each context v of $\mathcal{D}$, hypergraph $\mathcal{H}_v$ contains the "unknown part" of each clause $K \sqsubseteq M \in \mathcal{O}$ for which all literals are possible at v; furthermore, condition (R2) ensures that, for each edge $\langle v, u, \exists R.A \rangle$ of $\mathcal{D}$, hypergraphs $\mathcal{H}_v$ and $\mathcal{H}_u$ contain all unknown literals that might be needed to apply the Pred rule; and finally, condition (R3) ensures that, for each query $q \in \mathcal{Q}$ and $v = \vartheta(q)$, hypergraph $\mathcal{H}_v$ contains the "unknown part" of q. Conditions (R4)–(R9) then essentially capture the intuition that each context v of $\mathcal{D}$ is replaced by a tree decomposition $\mathcal{T}_v$ of $\mathcal{H}_v$. Please note that, for each edge $\langle v, u, \exists R.A \rangle$ of $\mathcal{D}$, by condition (R2) and property (T1) of tree decompositions, there exists at least one pair of w and z satisfying condition (R7). Furthermore, for each query $q \in \mathcal{Q}$ and $v = \vartheta(q)$, context $\vartheta'(q)$ can be an arbitrary vertex $w \in \mathcal{V}_v$ satisfying $(K \cup M) \cap \mathsf{uknw}(v) \subseteq \mathcal{L}_v(w)$; due to condition (R3) and property (T1) of tree decompositions, such w is guaranteed to exist. Finally, please note that decomposition $\mathcal{D}$ can admit many ϵ-refinements; however, by the following theorem, if $\mathcal{D}$ is sound and admissible, then each ϵ-refinement of $\mathcal{D}$ is sound and admissible too.

Theorem 43. *For each sound, admissible, and ϵ-free decomposition $\mathcal{D}$ of $\mathcal{O}$ and $\mathcal{Q}$, each ϵ-refinement $\mathcal{D}'$ of $\mathcal{D}$ is also a sound and admissible decomposition of $\mathcal{O}$ and $\mathcal{Q}$.*

Proof. We first show that $\mathcal{D}'$ is sound. Consider arbitrary contexts $w, z \in \mathcal{V}$, and let $v, u \in \mathcal{V}$ be such that $w \in \mathcal{V}_v$ and $z \in \mathcal{V}_u$. By condition (R5), we have $\mathsf{core}'(w) = \mathsf{core}(v)$, $\mathsf{knw}'(w) = \mathsf{knw}(v)$, $\mathsf{core}'(z) = \mathsf{core}(u)$, and $\mathsf{knw}'(z) = \mathsf{knw}(u)$.

Consider an arbitrary $\exists R.A$-edge $\langle w, z, \exists R.A \rangle \in \mathcal{E}'$. By condition (R7), then $\mathcal{D}$ contains the edge $\langle v, u, \exists R.A \rangle \in \mathcal{E}$. Since $\mathcal{D}$ is sound, $\mathcal{O} \models \mathsf{core}(v) \sqcap \exists R.A \sqsubseteq \exists R.[\mathsf{core}(u) \sqcap A]$ holds; but then, $\mathcal{O} \models \mathsf{core}'(w) \sqcap \exists R.A \sqsubseteq \exists R.[\mathsf{core}'(z) \sqcap A]$ holds as well.

Consider an arbitrary ϵ-edge $\langle w, z, \epsilon \rangle \in \mathcal{E}'$. By condition (R6), we have $v = u$, which clearly implies $\mathsf{core}'(w) = \mathsf{core}'(z)$, as required.

Finally, since $\mathcal{D}$ is sound, we have $\mathcal{O} \models \mathsf{core}(v) \sqsubseteq A$ for each $A \in \mathsf{knw}(v)$; but then, $\mathcal{O} \models \mathsf{core}'(w) \sqsubseteq A$ for each $A \in \mathsf{knw}'(w)$ clearly holds as well.

We next show that $\mathcal{D}'$ is admissible. Clearly, $\mathcal{D}'$ refers only to the literals occurring in $\mathcal{O} \cup \mathcal{Q}$, so $\mathcal{D}'$ is a decomposition of $\mathcal{O}$ and $\mathcal{Q}$. Furthermore, for each context $v \in \mathcal{V}$ the following conditions are satisfied.

$$\mathsf{knw}'(\mathcal{V}_v) = \bigcup_{w \in \mathcal{V}_v} \mathsf{knw}'(w) = \mathsf{knw}(v) \tag{139}$$

$$\mathsf{poss}'(\mathcal{V}_v) = \bigcup_{w \in \mathcal{V}_v} \mathsf{poss}'(w) \subseteq \mathsf{poss}(v) \tag{140}$$

Finally, for each ϵ-component $\mathcal{W}$ of $\mathcal{D}'$, a context $v \in \mathcal{V}$ exists such that $\mathcal{W} = \mathcal{V}_v$; conversely, for each context $v \in \mathcal{V}$, set $\mathcal{V}_v$ is an ϵ-component of $\mathcal{D}$. We next check that $\mathcal{D}'$ satisfies all the admissibility conditions listed in Definition 36.

Epsilon conditions:

(Condition E1) This property follows immediately from (R6) and the fact that the edges in $\mathcal{E}_v$ are undirected.

(Condition E2) By (R6), for each ϵ-component $\mathcal{V}_v$ of $\mathcal{D}'$, we have that $\mathcal{D}'_{\mathcal{V}_v}$ is equal to the graph $\langle \mathcal{V}_v, \mathcal{E}_v \rangle$, and the latter, being a tree decomposition, is an undirected tree.

(Condition E3) Consider an arbitrary ϵ-component $\mathcal{V}_v$ of $\mathcal{D}'$ and literal $L \in \mathsf{poss}'(\mathcal{V}_v)$, and let $\Gamma := \{w \in \mathcal{V}_v \mid L \in \mathsf{poss}'(w)\}$. We have $L \in \mathsf{poss}(v)$ due to (140), so either $L \in \mathsf{knw}(v)$ or $L \in \mathsf{uknw}(v)$. In the former case, we have $L \in \mathsf{poss}'(w)$ for each $w \in \mathcal{V}_v$, so $\Gamma = \mathcal{V}_v$ and Γ is clearly ϵ-connected. In the latter case, we have $\Gamma = \{w \in \mathcal{V}_v \mid L \in \mathcal{L}_v(w)\}$, so Γ is ϵ-connected by property (T2) of tree decompositions.

Structural conditions:

(Condition S1) By (R5), for each $v \in \mathcal{V}$ and each $w \in \mathcal{V}_v$, we have $\mathsf{core}'(w) = \mathsf{core}(v)$ and $\mathsf{knw}'(w) = \mathsf{knw}(v) \subseteq \mathsf{poss}'(w)$. Furthermore, $\mathsf{core}(v) \subseteq \mathsf{knw}(v)$ holds since $\mathcal{D}$ satisfies (S1), so we have $\mathsf{core}'(w) \subseteq \mathsf{knw}'(w) \subseteq \mathsf{poss}'(w)$.

(Condition S2) Consider an arbitrary ϵ-component $\mathcal{V}_u$ of $\mathcal{D}'$, an arbitrary edge $\langle w, z, \exists R.A \rangle \in \mathcal{E}'$ such that $z \in \mathcal{V}_u$, and an arbitrary literal $\forall \mathsf{inv}(R).C \in \mathsf{poss}'(\mathcal{V}_u)$; furthermore, let $v \in \mathcal{V}$ be the context of $\mathcal{D}$ such that $w \in \mathcal{V}_v$. By condition (R7) and $\langle w, z, \exists R.A \rangle \in \mathcal{E}'$, we have

$$\langle v, u, \exists R.A \rangle \in \mathcal{E}, \qquad \delta^{\exists R.A}_{v,u} \cap \mathsf{uknw}(v) \subseteq \mathcal{L}_v(w), \qquad \text{and} \qquad \varrho^{\exists R.A}_{v,u} \cap \mathsf{uknw}(v) \subseteq \mathcal{L}_u(z). \tag{141}$$

Now, by property (140) and $\forall \mathsf{inv}(R).C \in \mathsf{poss}'(\mathcal{V}_u)$, we have $\forall \mathsf{inv}(R).C \in \mathsf{poss}(u)$; hence, by admissibility condition (S2) for $\mathcal{D}$, we have $C \in \mathsf{poss}(v)$. But then, we have $C \in \delta^{\exists R.A}_{v,u}$ and $\forall \mathsf{inv}(R).C \in \varrho^{\exists R.A}_{v,u}$; together with (141), these conditions imply (142), so $\mathcal{D}'$ clearly satisfies admissibility condition (S2).

$$C \in \mathcal{L}_v(w) \cup \mathsf{knw}(v) = \mathsf{poss}'(w) \qquad\qquad \forall \mathsf{inv}(R).C \in \mathcal{L}_u(z) \cup \mathsf{knw}(u) = \mathsf{poss}'(z) \tag{142}$$

(Condition S3) Consider an arbitrary ϵ-component $\mathcal{V}_v$ of $\mathcal{D}'$ and an arbitrary literal $\exists R.A \in \mathsf{poss}'(\mathcal{V}_v)$. By property (140), we have $\exists R.A \in \mathsf{poss}(v)$, so, by admissibility condition (S3) for $\mathcal{D}$, an $\exists R.A$-edge $\langle v, u, \exists R.A \rangle \in \mathcal{E}$ exists satisfying (143).

$$\mathsf{poss}(u) \supseteq \{A\} \cup \{B \mid \forall R.B \in \mathsf{poss}(v)\} \tag{143}$$

Since $\mathcal{T}_v$ and $\mathcal{T}_u$ are tree decompositions of $\mathcal{H}_v$ and $\mathcal{H}_u$, respectively, by property (R2) of Definition 42 and property (T1) of tree decompositions, contexts $w \in \mathcal{V}_v$ and $z \in \mathcal{V}_u$ exist such that the following two properties hold.

$$\delta^{\exists R.A}_{v,u} \cap \mathsf{uknw}(v) \subseteq \mathcal{L}_v(w) \tag{144}$$

$$\delta^{\exists R.A}_{v,u} \cap \mathsf{uknw}(u) \subseteq \mathcal{L}_u(z) \tag{145}$$

But then, by condition (R7), we have $\langle w, z, \exists R.A \rangle \in \mathcal{E}'$. We next show that this edge satisfies condition (S3)—that is, that

(i) $\mathsf{poss}'(z) \supseteq \{A\} \cup \{B \mid \forall R.B \in \mathsf{poss}'(\mathcal{V}_v)\}$, and

(ii) $\mathsf{poss}'(w) \supseteq \{\exists R.A\} \cup \{\forall R.B \mid \forall R.B \in \mathsf{poss}'(\mathcal{V}_v)\}$.

By Definition 42 we have $\exists R.A \in \delta^{\exists R.A}_{v,u}$ and $A \in \varrho^{\exists R.A}_{v,u}$; furthermore, for each $\forall R.B \in \mathsf{poss}'(\mathcal{V}_v)$, we have $\forall R.B \in \mathsf{poss}(v)$ by (140), and $B \in \mathsf{poss}(u)$ by (143), which imply $\forall R.B \in \delta^{\exists R.A}_{v,u}$ and $B \in \varrho^{\exists R.A}_{v,u}$. But then, by (144) and (145), we have

$$\begin{array}{rlll}
\{A\} \cup & \{B \mid \forall R.B \in \mathsf{poss}'(\mathcal{V}_v)\} \subseteq \mathcal{L}_u(z) & \cup\ \mathsf{knw}(u) = \mathsf{poss}'(z) & \text{which proves (i), and} \\
\{\exists R.A\} \cup & \{\forall R.B \mid \forall R.B \in \mathsf{poss}'(\mathcal{V}_v)\} \subseteq \mathcal{L}_v(w) & \cup\ \mathsf{knw}(w) = \mathsf{poss}'(w) & \text{which proves (ii).}
\end{array}$$

Ontology condition:

Consider an arbitrary ϵ-component $\mathcal{V}_v$ of $\mathcal{D}'$ and an arbitrary clause $K \sqsubseteq M \in \mathcal{O}$ with $K \subseteq \mathsf{poss}'(\mathcal{V}_v)$. By (140), $K \subseteq \mathsf{poss}'(\mathcal{V}_v)$ implies $K \subseteq \mathsf{poss}(v)$, so by the ontology condition for $\mathcal{D}$ we have $M \subseteq \mathsf{poss}(v)$. By condition (R1) and property (T1) of tree decompositions, a context $w \in \mathcal{V}_v$ exists such that $(K \cup M) \cap \mathsf{uknw}(v) \subseteq \mathcal{L}_v(w)$. But then, $K \cup M \subseteq \mathcal{L}_v(w) \cup \mathsf{knw}(v) = \mathsf{poss}'(w)$ holds, so $\mathcal{D}'$ satisfies the ontology condition.

Ordering conditions:

(Condition P1) Consider an arbitrary $\exists R.A$-edge $\langle w, z, \exists R.A \rangle \in \mathcal{E}'$. By condition (R7), then $v, u \in \mathcal{V}$ exist such that $w \in \mathcal{V}_v$, $z \in \mathcal{V}_u$, and $\langle v, u, \exists R.A \rangle \in \mathcal{E}$. Since $\mathcal{D}$ satisfies admissibility condition (P1), each literal $\forall \mathsf{inv}(R).C \in \mathbf{L}$ is $\prec_u$-minimal. But then, by condition (R8), each such $\forall \mathsf{inv}(R).C$ is also $\prec'_z$-minimal; hence, $\prec'_z$ is R-admissible.

(Condition P2) By condition (R8), for each ϵ-edge $\langle u, w, \epsilon \rangle \in \mathcal{E}'$, all literals in $\mathsf{poss}'(u) \cap \mathsf{poss}'(w)$ are $\prec'_w$-minimal.

Covering condition:

Consider an arbitrary query $q = K \sqsubseteq M \in \mathcal{Q}$, and let $v = \vartheta(q)$ and $w = \vartheta'(q)$. By the covering condition for $\mathcal{D}$, context v covers $K \sqsubseteq M$, so $\mathsf{core}(v) \subseteq K \subseteq \mathsf{poss}(v)$ and M is $\prec_v$-minimal. By condition (R9), we have $w \in \mathcal{V}_v$ and $(K \cup M) \cap \mathsf{uknw}(v) \subseteq \mathcal{L}_v(w)$. This and condition (R5) imply that

$$\mathsf{core}'(w) = \mathsf{core}(v) \subseteq K \subseteq \mathsf{knw}(v) \cup \mathcal{L}_v(w) = \mathsf{poss}'(w).$$

Furthermore, by (140) and condition (R5), we have

$$M \cap \mathsf{poss}'(\mathcal{V}_v) \subseteq M \cap \mathsf{poss}(v) = M \cap [\mathsf{knw}(v) \cup \mathsf{uknw}(v)] \subseteq \mathsf{knw}(v) \cup [M \cap \mathsf{uknw}(v)] \subseteq \mathsf{knw}(v) \cup \mathcal{L}_v(w) = \mathsf{poss}'(w).$$

Finally, by condition (R8), disjunction M is $\prec'_w$-minimal because M is $\prec_v$-minimal. Thus, context w covers $K \sqsubseteq M$ in $\mathcal{D}'$. □

Definition 42 can be straightforwardly turned into an algorithm for computing an ϵ-refinement $\mathcal{D}'$ of $\mathcal{D}$: first, we compute all hypergraphs satisfying conditions (R1)–(R3); second, we compute a tree decomposition $\mathcal{T}_v$ for each hypergraph $\mathcal{H}_v$; third, we construct decomposition $\mathcal{D}'$ so that conditions (R4)–(R9) are satisfied; and fourth, for each context v of $\mathcal{D}$ and each $w \in \mathcal{V}_v$, we define $\prec'_w$ by modifying $\prec_v$ to ensure condition (R8). As explained earlier, in condition (R9) we can don't-care nondeterministically choose $\vartheta(q)$, and at least one suitable context is guaranteed to exist. Such an algorithm is fixed-parameter tractable, as shown by the following theorem.

Theorem 44. *For each integer* mwd*, one can compute an ϵ-refinement $\mathcal{D}'$ of $\mathcal{D}$ with* $\mathsf{wd}(\mathcal{D}') \leq \mathsf{mwd}$*, or determine that such $\mathcal{D}'$ does not exist, in time polynomial in* $f(\mathsf{mwd})$, $\mathsf{ln}(\mathcal{D})$*, and* $||\mathcal{O}|| + ||\mathcal{Q}||$*, for f a computable function.*

Proof. The $\mathbf{L}$-hypergraphs $\mathcal{H}_v$ corresponding to conditions (R1)–(R3) of Definition (42) can clearly be constructed in time polynomial in $\mathsf{ln}(\mathcal{D})$ and $||\mathcal{O}|| + ||\mathcal{Q}||$.

An ϵ-refinement $\mathcal{D}'$ with $\mathsf{wd}(\mathcal{D}') \leq \mathsf{mwd}$ exists if and only if, for each $v \in \mathcal{V}$, hypertraph $\mathcal{H}_v$ admits a tree decomposition $\mathcal{T}_v$ with $\mathsf{wd}(\mathcal{T}_v) \leq \mathsf{mwd}$. For each $v \in \mathcal{V}$, all literals occurring in $\mathcal{N}_v$ are contained in $\mathcal{O} \cup \mathcal{Q}$, so by Lemma 1, computing one $\mathcal{T}_v$ with $\mathsf{wd}(\mathcal{T}_v) \leq \mathsf{mwd}$, or determining that one does not exist can be done in time $O(f(\mathsf{mwd}) \cdot (||\mathcal{O}|| + ||\mathcal{Q}||))$.

Decomposition $\mathcal{D}'$ can be obtained from $\mathcal{D}$ by simply following conditions (R4)–(R8). Furthermore, for each query $q \in \mathcal{Q}$, one can define $\vartheta'(q)$ as an arbitrary element of $\mathcal{V}_{\vartheta(q)}$ satisfying condition (R9); due to condition (R3) and property (T1) of tree decomposition, such an element is guaranteed to exist. This construction is polynomial in $\mathsf{ln}(\mathcal{D}')$ and $||\mathcal{O}|| + ||\mathcal{Q}||$. Since the length of each $\mathcal{D}_v$ is bounded by the time it takes to compute it, the length of $\mathcal{D}'$ is bounded by $O(\mathsf{ln}(\mathcal{D}) \cdot f(\mathsf{mwd}) \cdot (||\mathcal{O}|| + ||\mathcal{Q}||))$, so the entire construction runs in time polynomial in $f(\mathsf{mwd})$, $\mathsf{ln}(\mathcal{D})$, and $||\mathcal{O}|| + ||\mathcal{Q}||$. □

This allows us to generalize our result on fixed-parameter tractability of subsumption reasoning from Section 4.

Theorem 45. *For each control $\mathcal{C}$, the following problem is fixed-parameter tractable:*

- Inputs: *a normalized $\mathcal{ALCI}$ ontology $\mathcal{O}$ and a set of queries $\mathcal{Q}$*
- Parameter: *an integer* mwd
- Problem: *return "yes" if an ϵ-refinement $\mathcal{D}$ of the $\mathcal{C}$-decomposition of $\mathcal{O}$ and $\mathcal{Q}$ exists with* $\mathsf{wd}(\mathcal{D}) \leq \mathsf{mwd}$*, and if also $\mathcal{O} \models K \sqsubseteq M$ holds for each query $K \sqsubseteq M \in \mathcal{Q}$*

Proof. Immediate by Theorems 29, 43, and 44, and Propositions 40 and 41. □

6. Soundness- and Admissibility-Preserving Decomposition Transformations

In this section we present several soundness- and admissibility-preserving decomposition transformations that, under certain conditions, can delete edges and/or contexts from decompositions. These transformations can be useful in practice as they can reduce the number of inferences of the consequence-based algorithm; moreover, we use these transformations in Section 7 to establish an exponential upper bound on the decomposition size. Throughout this section, we fix a normalized $\mathcal{ALCI}$ ontology $\mathcal{O}$, a finite set of queries $\mathcal{Q}$, and a sound and admissible decomposition $\mathcal{D} = \langle \mathcal{V}, \mathcal{E}, \mathsf{core}, \mathsf{knw}, \mathsf{poss}, \prec, \vartheta \rangle$ of $\mathcal{O}$ and $\mathcal{Q}$.

Definition 46 introduces a notion of redundancy for ϵ-edges, as well as a notion of ϵ-edge contraction that extends edge contraction from graph theory: given an edge $\langle v_1, v_2, \epsilon\rangle$ in $\mathcal{D}$, we merge contexts v_1 and v_2 into a new context v, eliminate the self-loop from v to v in order to preserve the tree shape of the enclosing ϵ-component, and define the core, known, and possible literals and the literal ordering of v to reflect both v_1 and v_2. Lemma 47 captures the properties of ϵ-edge contraction.

Definition 46. *An ϵ-edge $\langle v_1, v_2, \epsilon\rangle \in \mathcal{E}$ is* redundant *in $\mathcal{D}$ if $\mathsf{uknw}(v_1) \subseteq \mathsf{poss}(v_2)$. The result of* contracting *a (not necessarily redundant) ϵ-edge $\langle v_1, v_2, \epsilon\rangle \in \mathcal{E}$ is obtained as follows.*

(1) Remove v_1 and v_2 from $\mathcal{V}$, and add a fresh context v to $\mathcal{V}$.

(2) Replace each occurrence of v_1 and v_2 in ϑ and in $\mathcal{E}$ by v, and then remove the edge $\langle v, v, \epsilon\rangle$ from $\mathcal{E}$.

(3) Set $\mathsf{core}(v) := \mathsf{core}(v_1)$, $\mathsf{knw}(v) := \mathsf{knw}(v_1) \cup \mathsf{knw}(v_2)$, $\mathsf{poss}(v) := \mathsf{poss}(v_1) \cup \mathsf{poss}(v_2)$, and $\prec_v := \prec_{v_1} \cap \prec_{v_2}$.

Lemma 47. *Contracting an ϵ-edge preserves soundness and admissibility of a decomposition. Contracting a redundant ϵ-edge does not increase the width of a decomposition.*

Proof. It is straightforward to check that contracting an ϵ-edge preserves all conditions of Definitions 19 and 36. Assume now that a redundant ϵ-edge $\langle v_1, v_2, \epsilon\rangle \in \mathcal{E}$ is contracted. Then, $\mathsf{uknw}(v_1) \subseteq \mathsf{poss}(v_2)$ implies

$$\begin{aligned}\mathsf{uknw}(v) &= \mathsf{poss}(v) \setminus \mathsf{knw}(v) = [\mathsf{poss}(v_1) \cup \mathsf{poss}(v_2)] \setminus [\mathsf{knw}(v_1) \cup \mathsf{knw}(v_2)] \\ &\subseteq [[\mathsf{poss}(v_1) \setminus \mathsf{knw}(v_1)] \cup \mathsf{poss}(v_2)] \setminus \mathsf{knw}(v_2) = [\mathsf{uknw}(v_1) \cup \mathsf{poss}(v_2)] \setminus \mathsf{knw}(v_2) = \mathsf{poss}(v_2) \setminus \mathsf{knw}(v_2) = \mathsf{uknw}(v_2).\end{aligned}$$

Consequently, the width of a decomposition can only decrease as a result of contraction. □

Definition 48 specifies when a context w is broader than a context u in $\mathcal{D}$; roughly speaking, w is then "less specific" regarding core literals, but allows "more" possible literals. Given such w and u, we can then redirect each $\exists R.A$-edge ending at u so that the edge ends at w. That this operation preserves decomposition soundness and admissibility follows immediately from Lemma 49; moreover, this transformation manipulates only edges, so it clearly preserves decomposition width and length.

Definition 48. *A context $w \in \mathcal{V}$ of $\mathcal{D}$ is* broader *than a context $u \in \mathcal{V}$ of $\mathcal{D}$ if $\mathsf{core}(w) \subseteq \mathsf{core}(u)$, $\mathsf{poss}(w) \supseteq \mathsf{poss}(u)$, $\prec_w \subseteq \prec_u$, and $\mathsf{poss}(\mathcal{W}) \subseteq \mathsf{poss}(\mathcal{U})$, where $\mathcal{W}$ and $\mathcal{U}$ are the ϵ-components of $\mathcal{D}$ such that $w \in \mathcal{W}$ and $u \in \mathcal{U}$. For such w and u, the result of* redirecting *u to w in $\mathcal{D}$ is obtained by replacing each edge $\langle v, u, \exists R.A\rangle$ in $\mathcal{E}$ with $\langle v, w, \exists R.A\rangle$, and by replacing u with w in ϑ.*

Lemma 49. *Let $w, u \in \mathcal{V}$ be contexts of $\mathcal{D}$ such that w is broader than u. Then,*

1. *replacing an arbitrary $\exists R.A$-edge $\langle v, u, \exists R.A\rangle$ in $\mathcal{E}$ with $\langle v, w, \exists R.A\rangle$ preserves soundness and admissibility, and*
2. *each query q that is covered in u is also covered in w.*

Proof. (Claim 1) Assume that some $\langle v, u, \exists R.A\rangle$ in $\mathcal{E}$ is replaced with $\langle v, w, \exists R.A\rangle$. Only conditions (S2), (S3), and (P1) of Definition 36 concern $\exists R.A$-edges. To show that the "new" edge satisfies (S2), consider an arbitrary literal $\forall \mathsf{inv}(R).C \in \mathsf{poss}(\mathcal{W})$; then $\mathsf{poss}(\mathcal{W}) \subseteq \mathsf{poss}(\mathcal{U})$ implies $\forall \mathsf{inv}(R).C \in \mathsf{poss}(\mathcal{U})$; hence, by applying condition (S2) to the "old" edge $\langle v, u, \exists R.A\rangle$, we have $C \in \mathsf{poss}(v)$ and $\forall \mathsf{inv}(R).C \in \mathsf{poss}(u) \subseteq \mathsf{poss}(w)$, as required. Condition (S3) is clearly preserved due to $\mathsf{poss}(u) \subseteq \mathsf{poss}(w)$. Condition (P1) is preserved since $\prec_w \subseteq \prec_u$ and a subset of an R-admissible literal ordering is R-admissible.

(Claim 2) Consider an arbitrary query $K \sqsubseteq M$ that is covered in u; hence, $\mathsf{core}(u) \subseteq K \subseteq \mathsf{poss}(u)$, $M \cap \mathsf{poss}(\mathcal{U}) \subseteq \mathsf{poss}(u)$, and M is $\prec_u$-minimal. Since context w is broader than context u, we have $\mathsf{core}(w) \subseteq K \subseteq \mathsf{poss}(w)$, $M \cap \mathsf{poss}(\mathcal{W}) \subseteq \mathsf{poss}(w)$, and M is $\prec_w$-minimal; but then, $K \sqsubseteq M$ is also covered in w. □

Definition 50 introduces a notion of redundancy for $\exists R.A$-edges. Intuitively, an $\exists R.A$-edge $\langle v, u, \exists R.A\rangle$ is redundant in $\mathcal{D}$ if either $\exists R.A$ is not possible in the ϵ-component of v so condition (S3) of Definition 36 is satisfied vacuously, or $\mathcal{D}$ contains another $\exists R.A$-edge that satisfies condition (S3) for $\exists R.A$. It is obvious that deleting a redundant $\exists R.A$-edge in $\mathcal{D}$ preserves soundness and admissibility; moreover, doing so can be particularly useful if $\mathcal{D}$ is an ϵ-refinement of an ϵ-free decomposition. In particular, for each $v, u \in \mathcal{V}$ in condition (R7) of Definition 42, decomposition $\mathcal{D}$ then contains an $\exists R.A$-edge from each $w \in \mathcal{V}_v$ to each $z \in \mathcal{V}_u$ that satisfy $\delta_{v,u}^{\exists R.A} \cap \mathsf{uknw}(v) \subseteq \mathcal{L}_v(w)$ and $\varrho_{v,u}^{\exists R.A} \cap \mathsf{uknw}(u) \subseteq \mathcal{L}_u(z)$; however, only one such edge may be needed to satisfy admissibility condition (S3), so deleting the redundant $\exists R.A$-edges might be beneficial in practice.

Definition 50. *An $\exists R.A$-edge $\langle v, u, \exists R.A\rangle \in \mathcal{E}$ is* redundant *in $\mathcal{D}$ if either $\exists R.A \notin \mathsf{poss}(\mathcal{W})$, or another $\exists R.A$-edge $\langle w, z, \exists R.A\rangle \in \mathcal{E}$ with $w \in \mathcal{W}$ exists that satisfies condition* (S3) *of Definition 36 for $\exists R.A \in \mathsf{poss}(\mathcal{W})$, where $\mathcal{W}$ is the ϵ-component containing v.*

Definition 51 says that an entire ϵ-component $\mathcal{U}$ of $\mathcal{D}$ may be redundant if, for each context $u \in \mathcal{U}$, either (i) there is no $\exists R.A$-edge from another ϵ-component to u, and u is not used to cover a query in $\mathcal{Q}$, or (ii) u can be redirected to a context in another ϵ-component. In such a case, we can redirect appropriately all contexts satisfying condition (ii) to obtain a decomposition in which no context in $\mathcal{U}$ has an incoming $\exists R.A$-edge from another ϵ-component; but then, we can clearly delete all contexts in $\mathcal{U}$ and all the related edges without affecting decomposition admissibility.

Definition 51. *An ϵ-component $\mathcal{U}$ of $\mathcal{D}$ is* redundant *in $\mathcal{D}$ if at least one of the following holds for each context $u \in \mathcal{U}$:*

- *$v \in \mathcal{U}$ for each $\exists R.A$-edge $\langle v, u, \exists R.A \rangle \in \mathcal{E}$, and $\vartheta(q) \neq u$ for each query $q \in \mathcal{Q}$; or*
- *a context $w \in \mathcal{V} \setminus \mathcal{U}$ exists that is broader than u.*

A result of eliminating *such $\mathcal{U}$ from $\mathcal{D}$ is obtained from $\mathcal{D}$ as follows.*

1. *For each context $u \in \mathcal{U}$ such that either there exists an edge $\exists R.A$-edge $\langle v, u, \exists R.A \rangle \in \mathcal{E}$ with $v \notin \mathcal{U}$, or there exists a query $q \in \mathcal{Q}$ with $\vartheta(q) = u$, redirect u to some context $w \in \mathcal{V} \setminus \mathcal{U}$ that is broader than u.*

2. *Remove from the result of the previous step all contexts in $\mathcal{U}$ and all edges that are incident to a context in $\mathcal{U}$.*

Lemma 52. *Eliminating a redundant ϵ-component preserves soundness and admissibility, and it does not increase the width.*

Proof. By Lemma 49, step 1 preserves soundness and admissibility. Furthermore, by the definition of when $\mathcal{U}$ is redundant in $\mathcal{D}$, after step 1 each context $u \in \mathcal{U}$ neither occurs in an $\exists R.A$-edge $\langle v, u, \exists R.A \rangle$ with $v \notin \mathcal{U}$, nor is it used to cover a query from $\mathcal{Q}$; hence, step 2 preserves decomposition admissibility. Furthermore, removal of edges and contexts clearly preserves decomposition soundness. Finally, steps 1 and 2 clearly do not increase the width of a decomposition. □

Please note that context redirection, elimination of $\exists R.A$-edges, and elimination of redundant ϵ-components also apply to context structures (with obvious modifications), and they can be applied during execution of Algorithm 11. In this way, we obtain "structural" redundancy elimination rules that can further improve the performance of the consequence-based algorithm.

7. Bounds on Decomposition Length

In this section we discuss the bounds on the length of decompositions of minimum width. In particular, in Section 7.1 we consider the lower bound, and in Section 7.2 we turn our attention to the upper bound.

7.1. Lower Bound

We now exhibit a family of ontologies $\{\mathcal{O}_n\}$ and a fixed set of queries $\mathcal{Q}$ such that each sound and admissible decomposition of $\mathcal{O}_n$ and $\mathcal{Q}$ of minimum width necessarily has exponential length. Intuitively, this is because each ontology $\mathcal{O}_n$ has a "canonical" model $\mathcal{I}$ containing exponentially many domain elements, each of which satisfies a distinct combination of atomic concepts. Therefore, to obtain a decomposition of $\mathcal{O}_n$ and $\mathcal{Q}$ of minimal width, we must introduce a context for each domain element of $\mathcal{I}$, and thus such a decomposition will actually reflect the structure of $\mathcal{I}$. This is interesting because it shows that, in general, one cannot minimize decomposition width and still expect to obtain a practically useful decomposition (i.e., a decomposition of polynomial size). We use this observation as a justification for our practical approach to decomposition construction that we presented in Sections 4.4 and 5.3.

Theorem 53. *Let $\mathcal{Q} = \{C_0 \sqsubseteq \bot\}$. A family of $\mathcal{ALCI}$ ontologies $\{\mathcal{O}_n\}$ exists such that, for each $\mathcal{O}_n$,*

1. *a sound, admissible, and ϵ-free decomposition of $\mathcal{O}_n$ and $\mathcal{Q}$ of width 0 exists, and*

2. *each sound and admissible decomposition of $\mathcal{O}_n$ and $\mathcal{Q}$ of width 0 has length at least exponential in $\|\mathcal{O}_n\|$.*

For readability, we split the proof of Theorem 53 into several claims. Let $\mathcal{Q} = \{C_0 \sqsubseteq \bot\}$, let n be a positive integer, and let $\mathcal{O}_n$ be the ontology (of size linear in n) containing the following axioms for each $1 \leq i \leq n$.

$$C_{i-1} \sqsubseteq \exists R.C_i \quad (146) \qquad\qquad C_{i-1} \sqsubseteq \exists S.C_i \quad (150)$$

$$C_{i-1} \sqsubseteq \forall R.A_i \quad (147) \qquad\qquad C_{i-1} \sqsubseteq \forall S.B_i \quad (151)$$

$$A_i \sqsubseteq \forall R.A_i \quad (148) \qquad\qquad A_i \sqsubseteq \forall S.A_i \quad (152)$$

$$B_i \sqsubseteq \forall R.B_i \quad (149) \qquad\qquad B_i \sqsubseteq \forall S.B_i \quad (153)$$

Let $\mathbf{L} = \{C_0\} \cup \{C_i, A_i, B_i, \exists R.C_i, \exists S.C_i, \forall R.A_i, \forall S.A_i, \forall R.B_i, \forall S.B_i \mid 1 \leq i \leq n\}$ be the set of literals that occur in $\mathcal{O}_n$ and $\mathcal{Q}$. An *ABC-number* is a set of atomic concepts of the form $X = \{X_1, \ldots, X_k, C_k\}$, where $0 \leq k \leq n$ and each X_i is either A_i or B_i. By a slight abuse of notation, we often treat X as the conjunction of the atomic concepts contained in X; furthermore, the *rank* of X is k; finally, X^R and X^S are ABC-numbers of rank $k+1$ defined as

$$X^R := \{X_1, \ldots, X_k, A_{k+1}, C_{k+1}\} \qquad \text{and} \qquad X^S := \{X_1, \ldots, X_k, B_{k+1}, C_{k+1}\}.$$

Note that $\mathcal{O}_n \models X \sqsubseteq \exists R.X^R$ and $\mathcal{O}_n \models X \sqsubseteq \exists S.X^S$. Finally, for each ABC-number X, we define the set $\Gamma(X)$ of told subsumees of X w.r.t. $\mathcal{O}_n$ as follows:

$$\Gamma(X) := X \cup \{L \in \mathbf{L} \mid \text{an atomic concept } D \in X \text{ exists such that } D \sqsubseteq L \in \mathcal{O}_n\}$$

One can readily check that $\mathcal{O}_n$ entails only told subsumptions—that is, for each ABC-number X and each $L \in \mathbf{L}$, we have

$$\mathcal{O}_n \models X \sqsubseteq L \text{ if and only if } L \in \Gamma(X); \tag{154}$$

$$\mathcal{O}_n \models X \sqsubseteq \exists R.L \text{ if and only if } L \in \Gamma(X^R); \tag{155}$$

$$\mathcal{O}_n \models X \sqsubseteq \exists S.L \text{ if and only if } L \in \Gamma(X^S). \tag{156}$$

The following claim proves the first item of Theorem 53.

Claim 54. *There exists a sound, admissible, and ϵ-free decomposition $\mathcal{D}$ of $\mathcal{O}_n$ and $\mathcal{Q}$ such that* $\mathsf{wd}(\mathcal{D}) = 0$.

Proof. Let $\mathcal{D} = \langle \mathcal{V}, \mathcal{E}, \mathsf{core}, \mathsf{knw}, \mathsf{poss}, \prec, \vartheta \rangle$ be defined as follows:

$$\begin{aligned}
\mathcal{V} &:= \{v_X \mid X \text{ is an ABC-number}\} \\
\mathsf{core}(v_X) &:= X \\
\mathsf{knw}(v_X) &:= \mathsf{poss}(v_X) := \Gamma(X) \\
\prec_{v_X} &:= \emptyset \\
\vartheta(C_0 \sqsubseteq \bot) &:= v_{\{C_0\}} \\
\mathcal{E} &:= \{\langle v_X, v_{X^R}, \exists R.C_{k+1} \rangle, \langle v_X, v_{X^S}, \exists S.C_{k+1} \rangle \mid \text{for each ABC-number } X \text{ of rank } k \text{ with } 0 \le k < n\}
\end{aligned}$$

Clearly, $\mathcal{D}$ is an ϵ-free decomposition of $\mathcal{O}_n$ and $\mathcal{Q}$. Furthermore, it is straightforward to check that $\mathcal{D}$ is sound. Finally, $\mathcal{D}$ satisfies all the admissibility conditions of Definition 20: conditions (S1) and (S2) hold trivially, the definition of $\mathcal{E}$ ensures that condition (S3) is satisfied, the definition of poss ensures that the ontology condition is satisfied, the trivial literal ordering $\prec_{v_X}$ always satisfies the ordering condition, and the query $C_0 \sqsubseteq \bot$ is covered in context $v_{\{C_0\}}$. Finally, since $\mathsf{knw}(v_X) = \mathsf{poss}(v_X)$ for each context $v_X \in \mathcal{V}$, we have $\mathsf{wd}(\mathcal{D}) = 0$, as required. □

To prove the second item of Theorem 53, let $\mathcal{D} = \langle \mathcal{V}, \mathcal{E}, \mathsf{core}, \mathsf{knw}, \mathsf{poss}, \prec, \vartheta \rangle$ be an arbitrary sound and admissible decomposition of $\mathcal{O}_n$ and $\mathcal{Q}$ such that $\mathsf{wd}(\mathcal{D}) = 0$. Every ϵ-edge in a decomposition of width 0 is redundant in the sense of Definition 46, so we can assume without loss of generality that $\mathcal{D}$ is ϵ-free. We prove in Claim 56 that, for each ABC-number X, there exists a context $v_X \in \mathcal{V}$ such that $\mathsf{poss}(v_X) = \Gamma(X)$. Now $\Gamma(X)$ is distinct for each ABC-number X, so $\mathcal{V}$ clearly contains at least 2^n contexts, which concludes the proof of Theorem 53. Towards proving Claim 56, we first prove the following auxiliary claim.

Claim 55. *Let $v \in \mathcal{V}$ be a context and let X be an ABC-number. If* $\mathsf{core}(v) \subseteq \Gamma(X)$ *and* $\mathsf{poss}(v) \supseteq X$, *then* $\mathsf{poss}(v) = \Gamma(X)$.

Proof. Let $v \in \mathcal{V}$ be a context, and let X be an ABC-number such that $\mathsf{core}(v) \subseteq \Gamma(X)$ and $\mathsf{poss}(v) \supseteq X$ hold. By the ontology condition of Definition 20, set $\mathsf{poss}(v)$ is closed under Γ; thus, $\mathsf{poss}(v) \supseteq X$ implies $\mathsf{poss}(v) \supseteq \Gamma(X)$. To show that $\mathsf{poss}(v) \subseteq \Gamma(X)$ holds as well, consider an arbitrary literal $L \in \mathsf{poss}(v)$. Since $\mathsf{wd}(\mathcal{D}) = 0$, we have $L \in \mathsf{knw}(v)$. But then, by soundness condition (N3) of Definition 19, we have $\mathcal{O}_n \models \mathsf{core}(v) \sqsubseteq L$. Furthermore, we have $\mathcal{O}_n \models X \sqsubseteq \Gamma(X)$ by property (154), and $\mathsf{core}(v) \subseteq \Gamma(X)$ clearly implies $\mathcal{O}_n \models \Gamma(X) \sqsubseteq \mathsf{core}(v)$. Together, all these observations imply that $\mathcal{O}_n \models X \sqsubseteq L$ holds, which, by property (154), implies $L \in \Gamma(X)$. Consequently, we have $\mathsf{poss}(v) = \Gamma(X)$, as required. □

Claim 56. *For each ABC-number X, a context $v_X \in \mathcal{V}$ exists such that* $\mathsf{poss}(v_X) = \Gamma(X)$.

Proof. The proof is by induction on the rank of X. For the base case, we have $X = \{C_0\}$. Since $X \sqsubseteq \bot \in \mathcal{Q}$, query $X \sqsubseteq \bot$ is covered in the context $v_X = \vartheta(X \sqsubseteq \bot)$, and so we have $\mathsf{core}(v_X) \subseteq X \subseteq \mathsf{poss}(v_X)$ by the definition of covering. But then, Claim 55 implies that $\mathsf{poss}(v_X) = \Gamma(X)$ holds, as required.

For the induction step, assume that the claim holds for each ABC-number of rank $k-1$, and consider an arbitrary ABC-number of the form $Y = \{X_1, \ldots, X_{k-1}, X_k, C_k\}$; furthermore, let $X = \{X_1, \ldots, X_{k-1}, C_{k-1}\}$. By the induction hypothesis, a context $v_X \in \mathcal{V}$ exists such that $\mathsf{poss}(v_X) = \Gamma(X)$. We now consider the two possible forms of X_k.

Assume that $X_k = A_k$, which implies $Y = X^R$. By applying admissibility condition (S3) of Definition 20 to an existential restriction $\exists R.C_k \in \Gamma(X) = \mathsf{poss}(v_X)$, an edge $\langle v_X, v_Y, \exists R.C_k \rangle \in \mathcal{E}$ exists such that

$$\mathsf{poss}(v_Y) \supseteq \{C_k\} \cup \{D \mid \forall R.D \in \mathsf{poss}(v_X)\} = X^R.$$

Soundness condition (N1) of Definition 19 implies that $\mathcal{O}_n \models \mathsf{core}(v_X) \sqcap \exists R.C_k \sqsubseteq \exists R.[\mathsf{core}(v_Y) \sqcap C_k]$. Now $\mathcal{O}_n \models X \sqsubseteq \Gamma(X)$ holds by property (154); furthermore, $[\mathsf{core}(v_X) \sqcap \exists R.C_k] \subseteq \mathsf{poss}(v_X) = \Gamma(X)$ holds by the definition of poss, which clearly implies $\mathcal{O}_n \models \Gamma(X) \sqsubseteq [\mathsf{core}(v_X) \sqcap \exists R.C_k]$; thus, we have $\mathcal{O}_n \models X \sqsubseteq \exists R.[\mathsf{core}(v_Y) \sqcap C_k]$; but then, $\mathsf{core}(v_Y) \subseteq \Gamma(X^R)$ holds by (155). Thus, for $Y = X^R$, we have $\mathsf{core}(v_Y) \subseteq \Gamma(X^R)$ and $\mathsf{poss}(v_Y) \supseteq X^R$, so Claim 55 implies $\mathsf{poss}(v_Y) = \Gamma(X^R) = \Gamma(Y)$, as required.

The analysis for the case $X_k = B_k$ is analogous to the one above, with the difference that role S is used instead of role R. □

7.2. Upper Bound

We now show that, for each ontology $\mathcal{O}$ and each set of queries $\mathcal{Q}$, a sound and admissible decomposition of $\mathcal{O}$ and $\mathcal{Q}$ exists that has minimum width and at most exponential length. Intuitively, this is because, given an arbitrary decomposition of minimal width, we can obtain the required decomposition as follows. First, we repeatedly eliminate redundant ϵ-edges; this ensures that each ϵ-component contains at most a polynomial number of contexts (cf. Lemma 57), so the number of distinct ϵ-components is exponential. Second, we eliminate "duplicates" by repeatedly eliminating redundant ϵ-components. We thus obtain a decomposition of $\mathcal{O}$ and $\mathcal{Q}$ of at most exponential length; since none of these transformations increases decomposition width, the resulting decomposition is of minimal width as well (cf. Theorem 58).

Lemma 57. *Let $\mathcal{O}$ be a normalized $\mathcal{ALCI}$ ontology, let $\mathcal{Q}$ be a finite set of queries, and let $\mathcal{D} = \langle \mathcal{V}, \mathcal{E}, \mathsf{core}, \mathsf{knw}, \mathsf{poss}, \prec, \vartheta \rangle$ be an admissible decomposition of $\mathcal{O}$ and $\mathcal{Q}$ that has no redundant ϵ-edges. Then, for each ϵ-component $\mathcal{W}$ of $\mathcal{D}$, we have $|\mathcal{W}| \leq \mathsf{max}(|U|, 1)$ for $U = \bigcup_{w \in \mathcal{W}} \mathsf{uknw}(w)$.*

Proof. Let $\mathcal{W}$ be an arbitrary ϵ-component of $\mathcal{D}$ and let $n = |\mathcal{W}|$. If there exists a context $v \in \mathcal{W}$ with $\mathsf{uknw}(v) = \emptyset$, since $\mathcal{D}$ has no redundant ϵ-edges, then v is the only context in $\mathcal{W}$, so $|\mathcal{W}| = 1$ and the lemma holds. Thus, in the rest of the proof, we assume that $\mathsf{uknw}(v) \neq \emptyset$ for each context $v \in \mathcal{W}$.

Let $v_1, \dots, v_n$ be an arbitrary ordering of $\mathcal{W}$ obtained by an arbitrary depth-first traversal of the tree $\mathcal{D}_{\mathcal{W}}$. We define a function $\lambda \colon U \to \{1, \dots, n\}$ by setting $\lambda(L) := \mathsf{min}\{i \mid L \in \mathsf{uknw}(v_i)\}$. We claim that λ is surjective—that is, for each $1 \leq i \leq n$, a literal L exists such that $\lambda(L) = i$. For $i = 1$, since $\mathsf{uknw}(v_1) \neq \emptyset$, a literal $L \in \mathsf{uknw}(v_1)$ exists such that $\lambda(L) = 1$. Consider now an arbitrary i with $1 < i \leq n$, and let v_j be the parent of v_i in the depth-first traversal. Since $\mathcal{D}$ has no redundant ϵ-edges, we have $\mathsf{uknw}(v_i) \not\subseteq \mathsf{poss}(v_j)$, so a literal L exists such that $L \in \mathsf{uknw}(v_i) \setminus \mathsf{poss}(v_j)$. Now if $\lambda(L) < i$, then $v_{\lambda(L)}$ is visited in the depth-first traversal before v_i; thus, the unique path between $v_{\lambda(L)}$ and v_i goes through v_j, so, by the connectedness condition (E3) of admissibility, we have $L \in \mathsf{poss}(v_j)$; however, this contradicts our choice of L. Consequently, we have $\lambda(L) = i$, and so λ is surjective from U onto $\{1, \dots, n\}$. But then, $|\mathcal{W}| = n \leq |U|$. □

Theorem 58. *Let $\mathcal{O}$ be a normalized $\mathcal{ALCI}$ ontology, let $\mathcal{Q}$ be a finite set of queries, and let $\mathbf{L}$ be the set of literals occurring in $\mathcal{O} \cup \mathcal{Q}$. For each sound and admissible decomposition $\mathcal{D}'$ of $\mathcal{O}$ and $\mathcal{Q}$, there exists a sound and admissible decomposition $\mathcal{D}$ of $\mathcal{O}$ and $\mathcal{Q}$ such that $\mathsf{wd}(\mathcal{D}) \leq \mathsf{wd}(\mathcal{D}')$ and $\mathsf{ln}(\mathcal{D}) \leq 2^{O(|\mathbf{L}|^2)}$.*

Proof. Let $\mathcal{D}'$ be an arbitrary sound and admissible decomposition of $\mathcal{O}$ and $\mathcal{Q}$, and let $\mathcal{D} = \langle \mathcal{V}, \mathcal{E}, \mathsf{core}, \mathsf{knw}, \mathsf{poss}, \prec, \vartheta \rangle$ be an arbitrary decomposition obtained from $\mathcal{D}'$ by applying the following (nondeterministic) transformations.

1. Set $\prec_v := \emptyset$ for each context $v \in \mathcal{V}$.

2. While there are redundant ϵ-edges, pick one such ϵ-edge and contract it.

3. While there are redundant ϵ-components, pick one such ϵ-component and eliminate it.

By Lemmas 47 and 52, steps 2 and 3 preserve soundness and admissibility, and they do not increase the width; furthermore, step 1 trivially satisfies the two properties as well. To conclude our proof, we next show that $\mathsf{ln}(\mathcal{D}) \leq 2^{O(|\mathbf{L}|^2)}$.

Due to step 2, decomposition $\mathcal{D}$ contains no redundant ϵ-edges; hence, by Lemma 57, the cardinality of each ϵ-component of $\mathcal{D}$ is bounded by $|\mathbf{L}|$. We define the *type* of an ϵ-component $\mathcal{W}$ of $\mathcal{D}$ to be the pair $\langle \mathsf{core}(\mathcal{W}), \{\mathsf{poss}(v) \mid v \in \mathcal{W}\} \rangle$. Now assume that $\mathcal{U}$ and $\mathcal{W}$ are two different ϵ-components of $\mathcal{D}$ with the same type; then, for each context $u \in \mathcal{U}$, we can find a context $w \in \mathcal{W}$ with $\mathsf{poss}(u) = \mathsf{poss}(w)$; such w is broader than u because

- $\prec_w = \emptyset = \prec_u$ by step 1,
- $\mathsf{core}(w) = \mathsf{core}(\mathcal{W}) = \mathsf{core}(\mathcal{U}) = \mathsf{core}(u)$ by soundness condition (N2), and
- $\mathsf{poss}(\mathcal{W}) = \bigcup_{v \in \mathcal{W}} \mathsf{poss}(v) = \bigcup_{v \in \mathcal{U}} \mathsf{poss}(v) = \mathsf{poss}(\mathcal{U})$.

Consequently, such $\mathcal{U}$ is redundant in $\mathcal{D}$; however, due to step 3, decomposition $\mathcal{D}$ contains no redundant ϵ-components, so we obtain a contradiction. Therefore, all distinct ϵ-components of $\mathcal{D}$ are of a different type.

Thus, we can bound the number of ϵ-components of $\mathcal{D}$ by computing the number of possible types. For each ϵ-component $\mathcal{W}$ of $\mathcal{D}$, set $\mathsf{core}(\mathcal{W})$ is a subset of $\mathbf{L}$, so there are at most $2^{|\mathbf{L}|}$ such subsets; furthermore, set $\{\mathsf{poss}(v) \mid v \in \mathcal{W}\}$ is a subset of $2^{\mathbf{L}}$ of cardinality at most $|\mathbf{L}|$, so the number of such subsets is bounded by

$$\sum_{k=0}^{|\mathbf{L}|} \binom{2^{|\mathbf{L}|}}{k} \leq \sum_{k=0}^{|\mathbf{L}|} (2^{|\mathbf{L}|})^k \leq (2^{|\mathbf{L}|})^{(|\mathbf{L}|+1)}.$$

Consequently, the number of ϵ-components in $\mathcal{D}$ is bounded by $2^{|\mathbf{L}|} \cdot (2^{|\mathbf{L}|})^{(|\mathbf{L}|+1)}$. Since each ϵ-component contains at most $|\mathbf{L}|$ contexts, we have $\mathsf{ln}(\mathcal{D}) \leq |\mathbf{L}| \cdot 2^{|\mathbf{L}|(|\mathbf{L}|+2)}$, as required. □

8. Decomposition Width and Length in Practice

In this section we discuss the results of our experiments, whose goal was to measure decomposition width and length of realistic ontologies using several different expansion strategies.

Our experiments are based on our ontology repository that contains a diverse set of ontologies, including standard benchmarks and numerous ontologies from the life sciences domain. Each ontology in the repository can be accessed using a unique five-digit identifier.[4] To obtain our test corpus, we focused mostly on those ontologies in the repository that contain more than 1000 axioms and are not expressed in $\mathcal{EL}$ or DL-Lite$_{horn}$: ontologies expressed in one of these two languages have a decomposition of polynomial length and zero width, so we did not consider such ontologies interesting. However, we also included several smaller "toy" ontologies, such as Pizza and Wine, which were developed to demonstrate various ontology modeling constructs. In this way we obtained a corpus of 44 ontologies shown in Table 6. Since our framework is applicable only to normalized $\mathcal{ALCI}$ ontologies, we further transformed each ontology as follows. First, we eliminated from the ontology all axioms that are not supported in $\mathcal{SHI}$. Second, we transformed the result into a normalized $\mathcal{SHI}$ ontology using a normalization procedure similar to the one presented by Motik et al. [14]. Third, we eliminated role inclusion and transitivity axioms as described in Section 2 to obtain a normalized $\mathcal{ALCI}$ ontology. For each test ontology, Table 6 shows (i) the number of terminological axioms in the original ontology, (ii) the number of atomic concepts, atomic roles, and axioms in the $\mathcal{SHI}$ ontology after elimination of unsupported axioms, and (iii) the number of literals, roles, and clauses in the final $\mathcal{ALCI}$ ontology after normalization; ontologies are grouped to match Table 7, and the rationale behind the grouping will be explained shortly. For each normalized ontology $\mathcal{O}$, we defined the set $\mathcal{Q}_{\mathcal{O}}$ to contain a query $A \sqsubseteq B$ for all pairs of atomic concepts A and B in $\mathcal{O}$ that were not introduced during normalization; thus, $\mathcal{Q}_{\mathcal{O}}$ contains all queries relevant for ontology classification.

We developed three controls, $\mathcal{C}_1$, $\mathcal{C}_2$, and $\mathcal{C}_3$, that use the cautious, refined cautious, and eager expansion strategies, respectively, described in Section 3.4.2. Each control $\mathcal{C}_i$ used $\mathsf{mln} = \infty$—that is, we did not restrict the number of contexts introduced by the strategies. Although this is not formally allowed in our presentation of the algorithm, it allowed us to see how the strategies perform when run freely, without being forced to reuse contexts. The maximum number of contexts that could be introduced by $\mathcal{C}_1$ and $\mathcal{C}_2$ was thus polynomial, but for $\mathcal{C}_3$ it was exponential in the ontology size. Finally, each $\mathcal{C}_i$ used an initialization function that, for each atomic concept A occurring, introduces a single context v_A with $\mathsf{core}(v_A) = \{A\}$ to cover all queries of the form $A \sqsubseteq B$. Please note that the expansion strategies of $\mathcal{C}_1$ and $\mathcal{C}_3$ can reuse these "initial" contexts, but the refined cautious expansion strategy of $\mathcal{C}_2$ cannot since a context of the form v_A is distinct from a context of the form v_A^R for some role R.

For each normalized ontology $\mathcal{O}$, we conducted the following experiments. First, for each $1 \leq i \leq 3$, we computed the $\mathcal{C}_i$-decomposition $\mathcal{D}_i$ of $\mathcal{O}$ and $\mathcal{Q}_{\mathcal{O}}$ using the decomposition construction algorithm from Section 4.4. Second, for each $1 \leq i \leq 3$, we computed the width of an ϵ-refinement $\mathcal{D}'_i$ of $\mathcal{D}_i$ using the approach from Section 5.3. We computed tree decompositions of the relevant hypergraphs using the TreeD library;[5] however, since our hypergraphs were large, we used a heuristic mode in which the library returns only an approximation of the treewidth, without the actual tree decomposition. Thus, we were able to establish only an upper bound on the width, but not the length of $\mathcal{D}'_i$; however, by Lemma 57, we know that the size of each ϵ-component of $\mathcal{D}'_i$ is at most $\mathsf{wd}(\mathcal{D}_i)$, so an upper bound on $\mathsf{ln}(\mathcal{D}'_i)$ is given by $\mathsf{ln}(\mathcal{D}_i) \cdot \mathsf{wd}(\mathcal{D}_i)$.

Table 7 summarizes the results of our experiments. For readability, the ontologies were grouped into three subtables: Table 7a contains those ontologies for which all three strategies produce decompositions of different widths; Table 7b contains those ontologies for which the cautious and the refined cautious strategies produce decompositions of equal widths; and Table 7c contains those ontologies for which the choice of a strategy does not affect decomposition width. Moreover, each subtable is split into two parts, where the bottom part contains ontologies for which $\mathsf{wd}(\mathcal{D}_3) = 0$: these are all Horn ontologies on which the decomposition construction algorithm with the eager strategy fully solves the reasoning problem. On the "go-anatomy-importer" ontology, the cautious and the refined cautious strategies produce decompositions of very large width, which the TreeD library was unable to process. On the "nif-gross" ontology, the eager strategy seems to produce a decomposition of very large length, and our decomposition construction algorithm ran out of memory after introducing about five million contexts.

Our experiments show that decomposition length, even with the eager strategy, is in most cases commensurate with the number of atomic concepts. This is the main reason why consequence-based algorithms perform well in practice: our test ontologies do not seem to incur a substantial amount of and-branching; moreover, unlike the hypertableau algorithms, consequence-based algorithms do not suffer from unnecessary and-branching. The only outlier is the "nif-gross" ontology, on which the eager strategy produces very large decompositions; the cautious strategy, however, produces a decomposition of reasonably small width. This suggests that the hybrid strategy from Section 3.4.2 might actually be very useful in practice.

The refined cautious strategy does not seem to provide much practical benefit over the cautious strategy: there are only ten ontologies (all shown in Table 7a) on which $\mathsf{wd}(\mathcal{D}_1) > \mathsf{wd}(\mathcal{D}_2)$ holds, and in all cases the difference between $\mathsf{wd}(\mathcal{D}_1)$ and $\mathsf{wd}(\mathcal{D}_2)$ is negligible. Furthermore, only four decompositions satisfy $\mathsf{wd}(\mathcal{D}'_1) > \mathsf{wd}(\mathcal{D}'_2)$—that is, in all other cases, computing an

[4] An ontology with ID XXXXX can be readily downloaded from `http://www.cs.ox.ac.uk/isg/ontologies/UID/XXXXX.owl`, and a general description of our repository is available at `http://www.cs.ox.ac.uk/isg/ontologies/`.

[5] `http://www.itu.dk/people/sathi/treed/`

Table 6: Statistics for test ontologies

		Original		Reduced to $\mathcal{SHI}$			Normalized		
Ontology	ID	logic	TBox	A	T	axioms	L	R	clauses
acgt	00001	$\mathcal{SROIQ}(\mathcal{D})$	5,457	1,750	247	4,981	3,851	417	9,039
fma-cons	00285	$\mathcal{ALCOIF}(\mathcal{D})$	123,090	41,646	139	116,270	85,576	243	121,833
go-anatomy-importer	00040	$\mathcal{SRIQ}$	130,480	58,193	209	130,411	177,934	356	279,928
nif-gross	00354	$\mathcal{SROIF}(\mathcal{D})$	6,630	4,042	62	6,580	6,126	79	9,493
uberon	00658	$\mathcal{SRIQ}$	23,644	7,838	82	23,575	42,960	118	77,373
envo-xp	00448	$\mathcal{SRIF}$	74,103	35,702	95	74,065	50,836	51	75,154
galen7	00792	$\mathcal{ALERIF}(\mathcal{D})$	45,260	28,447	964	44,531	341,544	1,610	370,519
go-xp-all	00483	$\mathcal{SRI}$	115,252	89,926	191	115,214	238,923	319	295,029
pato	00762	$\mathcal{SHIF}$	7,616	2,307	24	2,820	1,737	34	2,641
plant-trait-xp	00573	$\mathcal{SRIF}$	127,603	60,968	96	127,559	85,831	46	130,235
cdao	00410	$\mathcal{SROIQ}(\mathcal{D})$	3,175	890	106	3,056	2,934	192	6,152
dolce-all	00024	$\mathcal{SHOIN}(\mathcal{D})$	1,544	204	313	1,502	10,679	626	16,442
ero	00450	$\mathcal{SHOIF}(\mathcal{D})$	3,277	2,397	103	3,146	3,965	189	5,075
gardiner-wafa	00055	$\mathcal{SHIN}$	246	152	20	242	277	27	317
influenza-ontology	00507	$\mathcal{SROIN}(\mathcal{D})$	1,552	745	66	1,504	3,739	116	5,963
lipid	00512	$\mathcal{ALCHIN}$	2,375	710	46	2,192	3,055	92	6,470
obi	00350	$\mathcal{SHOIN}(\mathcal{D})$	9,926	2,634	69	9,866	6,009	125	19,626
pizza	00793	$\mathcal{SHOIN}$	701	97	7	686	391	14	1,355
propreo	00772	$\mathcal{SHIN}$	565	474	30	548	994	48	1,086
sweet-space	00788	$\mathcal{SHOIN}(\mathcal{D})$	2,430	1,514	123	2,132	2,774	210	3,658
sweet-numerics	00789	$\mathcal{SHOIN}(\mathcal{D})$	2,499	1,499	137	2,185	2,768	239	3,599
sweet-phenomena	00790	$\mathcal{SHOIN}(\mathcal{D})$	2,658	1,721	112	2,367	2,871	191	3,734
fly-anatomy	00460	$\mathcal{SRI}$	19,477	7,798	21	19,466	10,391	27	19,534
galen-doctored	00029	$\mathcal{ALEHIF}^{+}$	4,762	2,748	413	4,586	11,447	534	13,769
galen-undoctored	00032	$\mathcal{ALEHIF}^{+}$	5,005	2,748	413	4,828	11,767	535	14,158
go	00764	$\mathcal{SRIF}$	76,286	36,495	13	76,279	82,191	22	123,506
worm-phenotype-xp	00675	$\mathcal{SHIF}$	90,522	51,413	113	90,497	74,594	95	100,966
aeo	00002	$\mathcal{SHIF}(\mathcal{D})$	20,317	759	47	3,372	1,487	92	5,598
mammalian-phenotype-xp	00518	$\mathcal{SR}$	6,684	9,809	24	6,680	32,111	48	38,994
nci-thesaurus	00554	$\mathcal{SH}(\mathcal{D})$	127,739	91,226	123	127,729	157,186	244	226,658
nif-cell	00352	$\mathcal{SROIF}(\mathcal{D})$	3,482	2,770	61	3,431	3,212	62	4,366
pharmacogenomics	00566	$\mathcal{SHOIN}(\mathcal{D})$	52,813	45,000	201	52,616	48,056	279	53,797
protein	00791	$\mathcal{S}$	37,266	35,196	7	37,266	46,240	10	61,424
sao	00624	$\mathcal{SHIN}(\mathcal{D})$	2,802	764	36	2,608	1,231	58	4,595
sct-sep	00778	$\mathcal{SH}$	54,978	54,974	9	54,977	333,067	13	503,379
software	00636	$\mathcal{SHOIQ}(\mathcal{D})$	2,661	903	27	2,501	1,721	31	3,064
vaccine	00668	$\mathcal{SRIQ}$	12,516	5,410	137	12,470	12,412	201	20,730
wine	00783	$\mathcal{SHOIN}(\mathcal{D})$	395	98	17	159	302	30	388
bams-simplified	00004	$\mathcal{SHIF}$	18,822	1,110	12	18,818	4,413	14	19,575
iedm	00293	$\mathcal{ALUN}(\mathcal{D})$	4,310	395	888	1,877	4,821	1,771	4,532
molecular-function-xp	00533	$\mathcal{SRIF}$	84,408	42,165	82	84,368	73,981	20	106,124
neuron-ontology	00006	$\mathcal{ALEHI}^{+}$	1,260	1,119	25	1,245	1,419	1	1,208
sequence	00627	$\mathcal{SHI}$	2,812	2,136	24	3,029	3,179	26	4,074
world-fact-book	00104	$\mathcal{ALCHOI}(\mathcal{D})$	1,158	1,080	4	1,122	1,090	1	1,120

Table 7: Decomposition width and length for test ontologies

	cautious			$\mathsf{cautious}_R$			eager		
Ontology	$\mathsf{ln}(\mathcal{D}_1)$	$\mathsf{wd}(\mathcal{D}_1)$	$\mathsf{wd}(\mathcal{D}'_1)$	$\mathsf{ln}(\mathcal{D}_2)$	$\mathsf{wd}(\mathcal{D}_2)$	$\mathsf{wd}(\mathcal{D}'_2)$	$\mathsf{ln}(\mathcal{D}_3)$	$\mathsf{wd}(\mathcal{D}_3)$	$\mathsf{wd}(\mathcal{D}'_3)$
(a) $\mathsf{wd}(\mathcal{D}_1)$, $\mathsf{wd}(\mathcal{D}_2)$, and $\mathsf{wd}(\mathcal{D}_3)$ are all different:									
acgt	1,754	235	11	1,883	151	9	1,851	162	9
fma-cons	41,646	418	22	80,606	389	22	84,647	387	22
go-anatomy-importer	58,192	12,909	?	79,533	12,582	?	94,915	491	28
nif-gross	4,069	686	178	4,653	589	152	$> 5 \cdot 10^6$	?	?
uberon	7,383	4,931	369	10,966	4,600	115	10,241	199	18
envo-xp	35,702	2	1	50,247	1	1	35,704	0	0
galen7	30,472	3,850	47	43,521	3,833	47	49,344	0	0
go-xp-all	89,926	1,456	76	107,797	1,404	49	92,801	0	0
pato	2,306	2	1	2,331	1	1	2,308	0	0
plant-trait-xp	60,968	2	1	84,271	1	1	60,973	0	0
(b) $\mathsf{wd}(\mathcal{D}_1) = \mathsf{wd}(\mathcal{D}_2) > \mathsf{wd}(\mathcal{D}_3)$ and $\mathsf{wd}(\mathcal{D}'_1) = \mathsf{wd}(\mathcal{D}'_2) \geq \mathsf{wd}(\mathcal{D}'_3)$:									
cdao	890	369	90	1,203	369	90	1,449	82	25
dolce-all	236	2,180	72	336	2,180	72	453	1,819	62
influenza-ontology	842	554	32	1,032	554	32	1,060	237	12
lipid	860	1,010	186	983	1,010	186	3,561	1,008	186
obi	2,826	293	39	3,249	293	39	3,287	233	17
pizza	101	101	34	147	101	34	392	41	10
propreo	478	147	22	483	147	22	561	75	13
sweet-space	1,517	60	6	1,522	60	6	1,540	38	5
ero	2,399	100	11	2,479	100	11	2,456	99	11
gardiner-wafa	158	34	6	169	34	6	1167	33	6
sweet-numerics	1,503	44	5	1,508	44	5	1,518	34	5
sweet-phenomena	1,723	44	5	1,728	44	5	1,739	43	5
fly-anatomy	7,798	5	2	10,356	5	2	7,873	0	0
galen-doctored	3,020	139	14	3,824	139	14	3,364	0	0
galen-undoctored	3,042	139	14	4,091	139	14	3,460	0	0
go	36,494	140	16	46,813	140	16	36,859	0	0
worm-phenotype-xp	51,413	6	2	66,320	6	2	51,421	0	0
(c) $\mathsf{wd}(\mathcal{D}_1) = \mathsf{wd}(\mathcal{D}_2) = \mathsf{wd}(\mathcal{D}_3)$ and $\mathsf{wd}(\mathcal{D}'_1) = \mathsf{wd}(\mathcal{D}'_2) = \mathsf{wd}(\mathcal{D}'_3)$:									
aeo	851	32	6	1,124	32	6	963	32	6
mammalian-phenotype-xp	9,809	16	5	13,668	16	5	9,809	16	5
nci-thesaurus	91,226	48	11	107,788	48	11	103,977	48	11
nif-cell	2,769	48	7	2,796	48	7	2,771	48	7
pharmacogenomics	45,015	321	26	46,844	321	26	45,348	321	26
protein	35,196	22	5	36,636	22	5	35,196	22	5
sao	769	108	9	843	108	9	856	108	9
sct-sep	54,973	2,602	278	76,545	2,602	278	54,973	2,602	278
software	1,120	9	5	1,422	9	5	1,234	9	5
vaccine	5,691	1,137	246	7,087	1,137	246	6,584	1,137	246
wine	121	122	27	122	122	27	698	122	27
bams-simplified	1,187	0	0	3,424	0	0	3,154	0	0
iedm	395	0	0	395	0	0	395	0	0
molecular-function-xp	42,165	0	0	60,140	0	0	42,165	0	0
neuron-ontology	1,119	0	0	1,419	0	0	1,119	0	0
sequence	2,136	0	0	2,446	0	0	2,136	0	0
world-fact-book	1,080	0	0	1,080	0	0	1,080	0	0

ϵ-refinement of $\mathcal{D}_1$ and $\mathcal{D}_2$ produces decompositions of equal width. The refined cautious strategy, however, can still be useful in practice: as we discussed in more detail in Section 3.4.2, contexts introduced by this strategy can have more refined literal orderings, but our framework cannot estimate the effects of a literal ordering on the performance of reasoning.

Particularly interesting are the results shown in the bottom part of Table 7c: all ontologies shown there contain universal restrictions and/or inverse roles, but they "behave" like $\mathcal{EL}$ ontologies in that there is no interaction between existential and universal restrictions and/or inverse roles; this is reflected by the fact that the cautious and the eager strategy introduce in most cases the same numbers of contexts. The only exception is the "bams-simplified" ontology, on which there is some interaction between existential and universal restrictions; however, even in this case, the cautious strategy can produce a decomposition of zero width. To understand how this can happen, consider the ontology $\{A \sqsubseteq \exists R.B,\ A \sqsubseteq \forall R.C,\ B \sqsubseteq C\}$: with the cautious strategy we get a context v_B with $\mathsf{core}(v_B) = \{B\}$, whereas with the eager strategy we get a context $v_{\{B,C\}}$ with $\mathsf{core}(v_{\{B,C\}}) = \{B, C\}$; however, in the former case C is a known consequence of B, so introducing a context with a "suboptimal" core does not increase decomposition width.

Finally, our experiments show that computing an ϵ-refinement can substantially reduce decomposition width: $\mathsf{wd}(\mathcal{D}'_i)$ is substantially lower than $\mathsf{wd}(\mathcal{D}_i)$ in all cases except when $\mathsf{wd}(\mathcal{D}_i)$ is already very small. Moreover, the eager strategy seems to produce decompositions of least width: with the exception of four ontologies ("dolce-all", "lipid", "sct-sep", and "vaccine"), decomposition $\mathcal{D}'_3$ has width below 30, and often much less than 30. This suggests that our approach to reducing or-branching from Section 5 could improve the performance of consequence-based reasoners in practice.

We finish this section with the observation that, in all cases, decomposition width is substantially smaller than the number of literals in the ontology, so our results provide a tighter estimate of the algorithm's complexity than that obtained from the usual complexity arguments. This is particularly true for Horn ontologies: decomposition length on such ontologies is manageable and decomposition width is zero, which explains why such ontologies are generally easier to reason with. On non-Horn ontologies, however, the upper complexity bound obtained using our results may still be very large; for example, for a decomposition of width 30, the number of clauses derived in each context is bounded by $4^{30} \approx 1.15 \cdot 10^{18}$—clearly, no algorithm deriving that many clauses can be deemed practical. This bound, however, only analyzes what could happen in the worst case and, similarly to propositional resolution, such worst cases are unlikely to occur in practice. We observed that decomposition widths of our test ontologies are mainly determined by universal restrictions involving inverse roles: in order to guarantee that all clauses needed to apply the Pred rule are derived, sets $\varrho_{v,u}^{\exists R.A}$ are needed in condition (R7) of Definition 42, and these sets can be large. In our experience, however, clauses with many literals of the form $\forall\mathsf{inv}(R).C$ are rarely derived during reasoning, which is why most ontologies can still be handled in practice. Therefore, a small decomposition width provides an indication, but not a definite guarantee, that reasoning with a particular ontology is easy. In contrast, decomposition length measures directly the number of contexts constructed in the consequence-based algorithm, so length is generally a good measure of ontology hardness.

9. Conclusion

In this paper we presented a very general framework for consequence-based reasoning in description logics. Then, we introduced a notion of decomposition that allows us to quantify the effects of and- and or-branching during reasoning. Finally, we presented what we believe to be the first result on fixed-parameter tractability of description logic reasoning. We see several theoretical and practical challenges for our future work.

On the theoretical side, we will try to extend our results to logics with number restrictions and nominals and thus obtain a consequence-based algorithm that can handle all of OWL 2 DL. Number restrictions have already been successfully integrated into resolution-based procedures so, while not trivial, incorporating them into consequence-based algorithms should be possible. We se two main challenges towards that goal. First, we believe that the notion of a context core will have to include two sets of literals (representing the types of an individual and its predecessor in a model) and a set of roles (describing the connections between the two individuals); the reason for this is analogous to why the hypertableau algorithm must use pairwise instead of single blocking [14] when an ontology contains both number restrictions and inverse roles. Second, we believe that our clauses will have to represent disjunctive properties of an individual's successors, which one can likely achieve using a skolemized clausal form as in first-order theorem proving [31]. In contrast, the combination of nominals, number restrictions, and inverse roles is a significant source of complexity, and it is currently unclear whether and how this combination of constructs can be handled in a consequence-based framework.

On the practical side, our results suggest several significant improvements to existing consequence-based reasoners. First, context reuse may allow reasoners to handle ontologies on which the eager strategy introduces a large number of contexts. Second, redundancy elimination techniques may turn out to further reduce the search space. Third, although computing tree decompositions in advance may incur significant overhead, we believe that computing ϵ-refinements "on demand" for contexts with large widths has the potential to further reduce the number of clauses derived. We will incorporate these ideas into a consequence-based reasoner and try to determine the extent to which they optimize the performance of reasoning in practice.

Appendix A. Proof of Theorem 6

Throughout Appendix A, we fix a normalized $\mathcal{ALCI}$ ontology $\mathcal{O}$, an admissible context structure $\mathcal{D} = \langle \mathcal{V}, \mathcal{E}, \mathsf{core}, \prec \rangle$, and a clause system $\mathcal{S}$ for $\mathcal{D}$ such that no inference rule from Table 3 is applicable to $\mathcal{S}$; furthermore, we fix $\mathbf{L}$ to be the set of all literals that occur in $\mathcal{O}$, $\mathcal{D}$, or $\mathcal{S}$. We prove Theorem 6 by showing the following contrapositive claim: $\mathcal{O} \not\models K \sqsubseteq M$ holds for each query $K \sqsubseteq M$ for which a context $v \in \mathcal{V}$ exists such that

- v is complete for $K \sqsubseteq M$,
- $K \sqsubseteq L \mathrel{\hat{\in}} \mathcal{S}(v)$ for each $L \in K$, and
- $K \sqsubseteq M \mathrel{\hat{\notin}} \mathcal{S}(v)$.

To this end, we construct a model $\mathcal{I}$ of $\mathcal{O}$ that refutes each query satisfying the above conditions. The construction is organized as follows. Instead of directly constructing a model, for simplicity we construct a *pre-model*—a graph-like structure with nodes labeled by sets of literals. We introduce this notion in Appendix A.1, and we explain how to convert a pre-model into a model. In Appendix A.2 we show how to construct a propositional interpretation satisfying a set of clauses closed under hyperresolution. In Appendix A.3 we prove certain properties of our inference rules that will allow us to use the construction from Appendix A.2. Finally, in Appendix A.4 we use these propositional interpretations to construct the desired pre-model of $\mathcal{O}$.

Appendix A.1. Pre-Interpretations and Pre-Models

In this section we introduce the notions of a pre-interpretation and a pre-model of an ontology, which we use to simplify the presentation of the subsequent proofs.

Definition 59. *A set of literals J* satisfies *a clause $K \sqsubseteq M$, written $J \models K \sqsubseteq M$, if $K \subseteq J$ implies $M \cap J \neq \emptyset$; otherwise, J* refutes *$K \sqsubseteq M$, written $J \not\models K \sqsubseteq M$.*

A pre-interpretation *is a labeled graph $I = \langle \Delta, E, J \rangle$ where Δ is a nonempty set of nodes, $E \subseteq \Delta \times \Delta \times \Sigma_R$ is a set of role-labeled edges, and $J \colon \Delta \to 2^{\Sigma_L}$ is a labeling of nodes by sets of literals satisfying the following two conditions.*

- *($I_\exists$) For each node $x \in \Delta$ and each literal $\exists R.A \in J(x)$, an edge $\langle x, y, R \rangle \in E$ exists such that $A \in J(y)$.*
- *($I_\forall$) For each edge $\langle x, y, R \rangle \in E$ and all literals $\forall R.B$ and $\forall \mathsf{inv}(R).C$, we have that*
  - *$\forall R.B \in J(x)$ implies $B \in J(y)$, and*
  - *$\forall \mathsf{inv}(R).C \in J(y)$ implies $C \in J(x)$.*

Pre-interpretation $I = \langle \Delta, E, J \rangle$ satisfies *a clause $K \sqsubseteq M$, written $I \models K \sqsubseteq M$, if $J(x) \models K \sqsubseteq M$ for each node $x \in \Delta$; otherwise, I* refutes *$K \sqsubseteq M$, written $I \not\models K \sqsubseteq M$. Finally, I is a* pre-model *of $\mathcal{O}$ if I satisfies each (normal) clause in $\mathcal{O}$.*

Given a pre-interpretation $I = \langle \Delta, E, J \rangle$, we can construct an interpretation $\mathcal{I}$ for all atomic concepts $A \in \Sigma_A$ and all atomic roles $T \in \Sigma_T$ as follows.

$$\Delta^{\mathcal{I}} := \Delta \tag{A.1}$$

$$A^{\mathcal{I}} := \{x \mid A \in J(x)\} \tag{A.2}$$

$$T^{\mathcal{I}} := \{\langle x, y \rangle \mid \langle x, y, T \rangle \in E\} \cup \{\langle y, x \rangle \mid \langle x, y, T^{-} \rangle \in E\} \tag{A.3}$$

Interpretation $\mathcal{I}$ satisfies the following property for each literal $L \in \Sigma_L$ and each node $x \in \Delta$:

$$L \in J(x) \quad \text{implies} \quad x \in L^{\mathcal{I}}. \tag{A.4}$$

For atomic concepts, property (A.4) holds by the definition of $A^{\mathcal{I}}$; furthermore, for literals of the form $\exists R.A$ and $\forall R.A$, property (A.4) holds due to conditions ($I_\exists$) and ($I_\forall$), respectively. The converse of property (A.4), however, holds only for atomic concepts, which is why all clauses in $\mathcal{O}$ must have only atomic concepts in their antecedents, and all queries must have only atomic concepts in their consequents. We next show that the interpretation $\mathcal{I}$ obtained as specified above is indeed a model of $\mathcal{O}$.

Lemma 60. *Let $I = \langle \Delta, E, J \rangle$ be a pre-interpretation, and let $\mathcal{I}$ be the interpretation obtained from I as specified in* (A.1)–(A.3)*. If I satisfies a normal clause α, then $\mathcal{I} \models \alpha$. Furthermore, if I refutes a query q, then $\mathcal{I} \not\models q$.*

Proof. Let $\bigsqcap_{i=1}^{m} A_i \sqsubseteq \bigsqcup_{j=1}^{n} L_j$ be an arbitrary normal clause that is satisfied in I, and consider an arbitrary node $x \in \Delta$ such that $x \in A_i^{\mathcal{I}}$ for each $1 \leq i \leq m$. By (A.2), we have $A_i \in J(x)$ for each $1 \leq i \leq m$; but then, since I satisfies the clause by the assumption, we have $L_j \in J(x)$ for some $1 \leq j \leq n$; hence, we have $x \in L_j^{\mathcal{I}}$ for some $1 \leq j \leq n$ by (A.4). Since this holds for arbitrary $x \in \Delta$, we have $\mathcal{I} \models \bigsqcap_{i=1}^{m} A_i \sqsubseteq \bigsqcup_{j=1}^{n} L_j$.

Let $\bigsqcap_{i=1}^{m} L_i \sqsubseteq \bigsqcup_{j=1}^{n} A_j$ be an arbitrary query that is refuted in I. Thus, a node $x \in \Delta$ exists such that $L_i \in J(x)$ for each $1 \leq i \leq m$, and $A_j \notin J(x)$ for each $1 \leq j \leq n$. But then, $x \in L_i^{\mathcal{I}}$ for each $1 \leq i \leq m$ by (A.4), and $x \notin A_j^{\mathcal{I}}$ for each $1 \leq j \leq n$ by (A.2); consequently, we have $\mathcal{I} \not\models \bigsqcap_{i=1}^{m} L_i \sqsubseteq \bigsqcup_{j=1}^{n} A_j$. □

Corollary 61. *If a pre-model I of $\mathcal{O}$ exists that refutes a query $K \sqsubseteq M$, then $\mathcal{O} \not\models K \sqsubseteq M$.*

Appendix A.2. Constructing Literal Interpretations

In this section, we show how the sets of clauses $\mathcal{S}(v)$ and the corresponding literal orderings $\prec_v$ can be used to construct the sets $J(x)$ of a pre-model $I = \langle \Delta, E, J \rangle$ that satisfies $\mathcal{O}$ but refutes a query $K \sqsubseteq M$. Our construction is independent from the selected context v; therefore, in the first part of this section we consider just a single set of clauses $\mathcal{N}$.

Throughout Appendix A.2 we fix a literal ordering $\prec$, a set of clauses $\mathcal{N}$ over $\mathbf{L}$, and a clause $K \sqsubseteq M$ over $\mathbf{L}$ satisfying the following conditions:

$$M \text{ is } \prec\text{-minimal,} \tag{A.5}$$
$$K \sqsubseteq L \mathrel{\hat{\in}} \mathcal{N} \text{ for each literal } L \in K, \text{ and} \tag{A.6}$$
$$K \sqsubseteq M \mathrel{\hat{\notin}} \mathcal{N}. \tag{A.7}$$

Next, in Definition 62 we introduce a notion that determines when set $\mathcal{N}$ contains "sufficiently many" hyperresolution consequences w.r.t. literal ordering $\prec$.

Definition 62. *Set $\mathcal{N}$ is* closed under $\prec$-hyperresolution with a clause $\bigsqcap_{i=1}^{n} L_i \sqsubseteq M'$ *(not necessarily contained in $\mathcal{N}$) if, for each choice of n clauses $K_i \sqsubseteq M_i \sqcup L_i \in \mathcal{N}$, $1 \leq i \leq n$, each satisfying $L_i \not\prec M_i$, we have $\bigsqcap_{i=1}^{n} K_i \sqsubseteq M' \sqcup \bigsqcup_{i=1}^{n} M_i \mathrel{\hat{\in}} \mathcal{N}$.*

The following lemma follows trivially from Definition 62 and the notion of clause strengthening.

Lemma 63. *Let α_1 and α_2 be arbitrary clauses such that α_1 is a strengthening of α_2. If $\mathcal{N}$ is closed under $\prec$-hyperresolution with α_1, then $\mathcal{N}$ is also closed under $\prec$-hyperresolution with α_2.*

For $\prec$, $\mathcal{N}$, and $K \sqsubseteq M$ fixed as specified above, we next construct a set of literals J that refutes $K \sqsubseteq M$, but that satisfies each clause $\bigsqcap_{i=1}^{n} L_i \sqsubseteq M'$ for which $\mathcal{N}$ is closed under $\prec$-hyperresolution. We achieve this by essentially adapting the standard techniques from resolution theorem proving [31] to our setting. Towards this goal, we proceed as follows: we first present the construction of J; in Lemma 64 we prove certain properties of the productive clauses in $\mathcal{N}$; in Lemma 65 we show that J satisfies each clause $K' \sqsubseteq M'$ for which a strengthening is contained in $\mathcal{N}$ and that satisfies $K' \subseteq K$; in Lemma 66 we show that J refutes $K \sqsubseteq M$; and finally in Lemma 67 we prove the desired result.

Let $L_1, L_2, \ldots, L_\ell$ be an arbitrary total ordering of $\mathbf{L}$ that extends $\prec$ such that the elements of M precede all the remaining literals from $\mathbf{L}$—that is, for all i and j, we have that

$$L_i \prec L_j \text{ implies } i < j, \text{ and} \tag{A.8}$$
$$i < j \text{ and } L_j \in M \text{ imply } L_i \in M. \tag{A.9}$$

Since M is $\prec$-minimal by (A.5) and set $\mathbf{L}$ is finite, at least one such total ordering exists. For each $0 \leq k \leq \ell$, let $\mathbf{L}_k := \{L_1, \ldots, L_k\}$, and let J_k be an inductively defined set of literals such that $J_0 := \emptyset$ and, for each $0 < k \leq \ell$, we have

$$J_k := \begin{cases} J_{k-1} \cup \{L_k\} & \text{if a clause } K' \sqsubseteq M' \sqcup L_k \in \mathcal{N} \text{ exists such that } K' \subseteq K,\ M' \cap J_{k-1} = \emptyset, \text{ and } M' \subseteq \mathbf{L}_{k-1} \\ J_{k-1} & \text{otherwise.} \end{cases} \tag{A.10}$$

Let $J := J_\ell$. Each set J that can be obtained by such a construction (which is nondeterministic due to the choice of the total ordering) is called a *literal interpretation* of $\mathbf{L}$ w.r.t. $\mathcal{N}$ and $\prec$ refuting $K \sqsubseteq M$. Each clause $K' \sqsubseteq M' \sqcup L_k \in \mathcal{N}$ that satisfies the first condition in (A.10) for some k is said to be a *productive clause*, and the clause is said to *produce* the literal L_k in J. Clearly, each literal in J is produced by at least one productive clause in $\mathcal{N}$.

Lemma 64. *Each productive clause $K' \sqsubseteq M' \sqcup L_k \in \mathcal{N}$ satisfies $K' \subseteq K$, $M' \cap J = \emptyset$, and $L_k \not\prec M'$.*

Proof. Let $K' \sqsubseteq M' \sqcup L_k \in \mathcal{N}$ be an arbitrary clause producing literal L_k in J for some integer k. By (A.10), we have $K' \subseteq K$, $M' \cap J_{k-1} = \emptyset$, and $M' \subseteq \mathbf{L}_{k-1}$. Clearly $J_{k-1} = J \cap \mathbf{L}_{k-1}$, so we have $M' \cap J = \emptyset$. Finally, by (A.8), we have $L_k \not\prec L$ for each $L \in \mathbf{L}_{k-1}$; hence, due to $M' \subseteq \mathbf{L}_{k-1}$, we have $L_k \not\prec M'$. □

Lemma 65. *For each clause $K' \sqsubseteq M'$ such that $K' \sqsubseteq M' \mathrel{\hat{\in}} \mathcal{N}$ and $K' \subseteq K$, we have $M' \cap J \neq \emptyset$.*

Proof. Let $K' \sqsubseteq M'$ be an arbitrary clause satisfying $K' \sqsubseteq M' \mathrel{\hat{\in}} \mathcal{N}$ and $K' \subseteq K$. Then, a clause $K_1 \sqsubseteq M_1 \in \mathcal{N}$ exists such that $K_1 \subseteq K' \subseteq K$ and $M_1 \subseteq M'$ hold. Clause $K_1 \sqsubseteq \bot$ is thus a strengthening of $K \sqsubseteq M$, but $K \sqsubseteq M \mathrel{\hat{\notin}} \mathcal{N}$ by assumption (A.7); thus, $M_1 \neq \emptyset$. Let k be the largest integer for which $L_k \in M'$ holds; hence, clause $K_1 \sqsubseteq M_1$ is of the form $K_1 \sqsubseteq M_2 \sqcup L_k$ where $M_2 \subseteq \mathbf{L}_{k-1}$. The claim of this lemma holds trivially if $M_2 \cap J \neq \emptyset$. Furthermore, if $M_2 \cap J = \emptyset$, then by (A.10) clause $K_1 \sqsubseteq M_2 \sqcup L_k$ produces $L_k \in J$, so we again have $M' \cap J \neq \emptyset$, as required. □

Lemma 66. *We have $J \not\models K \sqsubseteq M$—that is, $K \subseteq J$ and $M \cap J = \emptyset$.*

Proof. ($K \subseteq J$) Consider an arbitrary literal $L \in K$. Then $K \sqsubseteq L \mathrel{\hat{\in}} \mathcal{N}$ by assumption (A.6), so a conjunction $K' \subseteq K$ exists such that $K' \sqsubseteq \bot \in \mathcal{N}$ or $K' \sqsubseteq L \in \mathcal{N}$. The former contradicts $K \sqsubseteq M \mathrel{\hat{\notin}} \mathcal{N}$, which is assumption (A.7); hence, we have $K' \sqsubseteq L \in \mathcal{N}$, so by (A.10) this clause produces $L \in J$.

($M \cap J = \emptyset$) Assume that a literal $L_k \in M \cap J$ exists, and let $K' \sqsubseteq M' \sqcup L_k \in \mathcal{N}$ be a clause that produces $L_k \in J$. By (A.10), we have $M' \subseteq \mathbf{L}_{k-1}$ and $K' \subseteq K$; furthermore, by (A.9) and $L_k \in M$, we have $M' \cup \{L_k\} \subseteq M$. But then, $K' \sqsubseteq M' \sqcup L_k \in \mathcal{N}$ contradicts $K \sqsubseteq M \mathrel{\hat{\notin}} \mathcal{N}$, which is assumption (A.7). □

Lemma 67. *For each clause α such that $\mathcal{N}$ is closed under $\prec$-hyperresolution with α, we have $J \models \alpha$.*

Proof. Assume that $\alpha = \bigsqcap_{i=1}^{n} L_i \sqsubseteq M'$ and that $L_i \in J$ for each $1 \leq i \leq n$. For each $1 \leq i \leq n$, let $K_i \sqsubseteq M_i \sqcup L_i$ be some clause that produces $L_i \in J$; then $K_i \subseteq K$, $M_i \cap J = \emptyset$, and $L_i \not\prec M_i$ by Lemma 64. Since $\mathcal{N}$ is closed under $\prec$-hyperresolution with α, we have $\bigsqcap_{i=1}^{n} K_i \sqsubseteq M' \sqcup \bigsqcup_{i=1}^{n} M_i \mathrel{\hat{\in}} \mathcal{N}$. By $(\bigsqcap_{i=1}^{n} K_i) \subseteq K$ and Lemma 65, we have $(M' \sqcup \bigsqcup_{i=1}^{n} M_i) \cap J \neq \emptyset$. Finally, since $M_i \cap J = \emptyset$ holds for each $1 \leq i \leq n$, we have $M' \cap J \neq \emptyset$. Consequently, we have $J \models \alpha$, as required. □

Appendix A.3. Properties of the Consequence-Based Inference Rules

To apply the construction from Appendix A.2 to our clause system $\mathcal{S}$ and the context structure $\mathcal{D} = \langle \mathcal{V}, \mathcal{E}, \mathsf{core}, \prec \rangle$ (which were fixed at the beginning of Appendix A), we next prove the following two auxiliary lemmas about the consequence-based inference rules. Lemma 68 allows us to show that, for each context $v \in \mathcal{V}$, the set of literals obtained by applying the above construction to $\mathcal{S}(v)$ satisfies each clause in $\mathcal{O}$; the lemma follows trivially from Definition 62 and the fact that the Hyper rule is not applicable to $\mathcal{S}$. Furthermore, Lemma 69 shows a similar property for the clauses that participate in the Pred rule.

Lemma 68. *For each context $v \in \mathcal{V}$ and each clause $\alpha \in \mathcal{O}$, set $\mathcal{S}(v)$ is closed under $\prec_v$-hyperresolution with α.*

Lemma 69. *For each edge $\langle v, u, \exists R.A \rangle \in \mathcal{E}$ and each clause $\alpha = A \sqcap \bigsqcap_{i=1}^{n} B_i \sqsubseteq \bigsqcup_{j=1}^{m} \forall \mathsf{inv}(R).C_j$ such that $\alpha \mathrel{\hat{\in}} \mathcal{S}(u)$, set $\mathcal{S}(v)$ is closed under $\prec_v$-hyperresolution with $\exists R.A \sqcap \bigsqcap_{i=1}^{n} \forall R.B_i \sqsubseteq \bigsqcup_{j=1}^{m} C_j$.*

Proof. Consider an arbitrary edge $\langle v, u, \exists R.A \rangle \in \mathcal{E}$ and an arbitrary clause $\alpha = A \sqcap \bigsqcap_{i=1}^{n} B_i \sqsubseteq \bigsqcup_{j=1}^{m} \forall \mathsf{inv}(R).C_j$ such that $\alpha \mathrel{\hat{\in}} \mathcal{S}(u)$. By the definition of $\hat{\in}$, a subset $\{B'_i\}_{i=1}^{n'}$ of $\{B_i\}_{i=1}^{n}$ and a subset $\{C'_j\}_{j=1}^{m'}$ of $\{C_j\}_{j=1}^{m}$ exist such that

$$A \sqcap \bigsqcap_{i=1}^{n'} B'_i \sqsubseteq \bigsqcup_{j=1}^{m'} \forall \mathsf{inv}(R).C'_j \in \mathcal{S}(u) \qquad \text{or} \qquad \bigsqcap_{i=1}^{n'} B'_i \sqsubseteq \bigsqcup_{j=1}^{m'} \forall \mathsf{inv}(R).C'_j \in \mathcal{S}(u).$$

The Pred rule is not applicable to clause system $\mathcal{S}$; thus, for each clause $K_0 \sqsubseteq M_0 \sqcup \exists R.A \in \mathcal{S}(v)$ satisfying $\exists R.A \not\prec_v M_0$, and for each choice of n' clauses $K_i \sqsubseteq M_i \sqcup \forall R.B'_i$, $1 \leq i \leq n'$, satisfying $\forall R.B'_i \not\prec_v M_i$, we have

$$K_0 \sqcap \bigsqcap_{i=1}^{n'} K_i \sqsubseteq M_0 \sqcup \bigsqcup_{i=1}^{n'} M_i \sqcup \bigsqcup_{j=1}^{m'} C'_j \mathrel{\hat{\in}} \mathcal{S}(v).$$

Thus, $\mathcal{S}(v)$ is closed under $\prec_v$-hyperresolution with clause $\exists R.A \sqcap \bigsqcap_{i=1}^{n'} \forall R.B'_i \sqsubseteq \bigsqcup_{j=1}^{m'} C'_j$. By Lemma 63, then $\mathcal{S}(v)$ is also closed under $\prec_v$-hyperresolution with clause $\exists R.A \sqcap \bigsqcap_{i=1}^{n} \forall R.B_i \sqsubseteq \bigsqcup_{j=1}^{m} C_j$ since the former is a strengthening of the latter clause. □

Appendix A.4. The Completeness Claim

We now complete the proof of Theorem 6 by constructing a pre-interpretation that satisfies $\mathcal{O}$ but refutes the relevant queries. To simplify the presentation, given a context $v \in \mathcal{V}$ and a clause $K \sqsubseteq M$, we write $v \nvdash K \sqsubseteq M$ if and only if

- M is $\prec_v$-minimal (i.e., v is complete for $K \sqsubseteq M$),
- $K \sqsubseteq L \mathrel{\hat{\in}} \mathcal{S}(v)$ for each literal $L \in K$, and
- $K \sqsubseteq M \mathrel{\hat{\notin}} \mathcal{S}(v)$.

Claim 70. *For each context $v \in \mathcal{V}$ and each clause $K \sqsubseteq M$, if $K \sqsubseteq L \mathrel{\hat{\in}} \mathcal{S}(v)$ for each $L \in K$ and $K \sqsubseteq M \mathrel{\hat{\notin}} \mathcal{S}(v)$, then $K \subseteq \mathbf{L}$.*

Proof. Assume that there exists a literal $L \in K \setminus \mathbf{L}$; then, L does not occur in $\mathcal{S}(v)$, so $K \sqsubseteq L \mathrel{\hat{\in}} \mathcal{S}(v)$ implies $K \sqsubseteq \bot \mathrel{\hat{\in}} \mathcal{S}(v)$, which contradicts the assumption that $K \sqsubseteq M \mathrel{\hat{\notin}} \mathcal{S}(v)$. Thus, $K \subseteq \mathbf{L}$ holds. □

Theorem 6 holds vacuously if no context $v \in \mathcal{V}$ and no query $K \sqsubseteq M$ exist such that $v \nvdash K \sqsubseteq M$. Thus, we assume that $v \nvdash K \sqsubseteq M$ holds for at least one context $v \in \mathcal{V}$ and at least one query $K \sqsubseteq M$; by the definition of clause strengthening, we then also have $v \nvdash K \sqsubseteq M'$ for $M' = M \cap \mathbf{L}$; the latter clause is over $\mathbf{L}$ by Claim 70. Let $I = \langle \Delta, E, J \rangle$ be defined as follows.

- Set Δ is the smallest set that contains a distinguished element $\delta^{v}_{K \sqsubseteq M}$ for each context $v \in \mathcal{V}$ and each clause $K \sqsubseteq M$ over $\mathbf{L}$ such that $v \nvdash K \sqsubseteq M$. Set Δ is not empty due to the above assumption.

- For each $\delta^v_{K \sqsubseteq M} \in \Delta$, let $J(\delta^v_{K \sqsubseteq M})$ be an arbitrary literal interpretation of $\mathbf{L}$ w.r.t. $\mathcal{S}(v)$ and $\prec_v$ refuting $K \sqsubseteq M$.
- Set E is the smallest set containing $\langle \delta^v_{K_1 \sqsubseteq M_1}, \delta^u_{A \sqcap K_2 \sqsubseteq M_2}, R \rangle$ for each element $\delta^v_{K_1 \sqsubseteq M_1} \in \Delta$, each literal $\exists R.A \in J(\delta^v_{K_1 \sqsubseteq M_1})$, and each edge $\langle v, u, \exists R.A \rangle \in \mathcal{E}$ such that $u \nvdash A \sqcap K_2 \sqsubseteq M_2$, where K_2 and M_2 are as specified below.

$$K_2 = \bigsqcap \{B \mid \forall R.B \in J(\delta^v_{K_1 \sqsubseteq M_1})\} \tag{A.11}$$

$$M_2 = \bigsqcup \{\forall \mathsf{inv}(R).C \in \mathbf{L} \mid C \notin J(\delta^v_{K_1 \sqsubseteq M_1})\} \tag{A.12}$$

In the last item, we have $K_2 \cup M_2 \subseteq \mathbf{L}$ and $A \in \mathbf{L}$; thus, $u \nvdash A \sqcap K_2 \sqsubseteq M_2$ ensures $\delta^u_{A \sqcap K_2 \sqsubseteq M_2} \in \Delta$, and so E is correctly defined. Next, in Claim 71 we show that I is a pre-interpretation, in Claim 72 we show that I is a pre-model of $\mathcal{O}$, and in Claim 73 we show that I refutes the relevant queries. The claim of Theorem 6 then follows from Claims 72 and 73, and Corollary 61.

Claim 71. *Structure I satisfies conditions $(I_\exists)$ and $(I_\forall)$ of pre-interpretations from Definition 59.*

Proof. (Property $I_\exists$) Consider an arbitrary element $\delta^v_{K_1 \sqsubseteq M_1} \in \Delta$ and an arbitrary literal $\exists R.A \in J(\delta^v_{K_1 \sqsubseteq M_1})$; we next show that an edge $\langle \delta^v_{K_1 \sqsubseteq M_1}, \gamma, R \rangle \in E$ exists such that $A \in J(\gamma)$. The Succ rule is not applicable to $\mathcal{S}$, so an edge $\langle v, u, \exists R.A \rangle \in \mathcal{E}$ exists satisfying

$$L \sqsubseteq L \mathrel{\hat{\in}} \mathcal{S}(u) \text{ for each } L \in \{A\} \cup \mathbf{B}_p, \text{ where } \mathbf{B}_p = \{B \mid K' \sqsubseteq M' \sqcup \forall R.B \in S(v) \text{ and } \forall R.B \not\prec_v M'\}. \tag{A.13}$$

Let K_2 and M_2 be as specified in (A.11) and (A.12). The following observations prove that $u \nvdash A \sqcap K_2 \sqsubseteq M_2$.

- Admissibility of $\mathcal{D}$ and $\langle v, u, \exists R.A \rangle \in \mathcal{E}$ imply, by Definition 4, that $\prec_u$ is R-admissible, so M_2 is $\prec_u$-minimal.
- Consider an arbitrary literal $L \in A \sqcap K_2$; we show that $A \sqcap K_2 \sqsubseteq L \mathrel{\hat{\in}} \mathcal{S}(u)$. Assume that $L \in K_2$, so L is an atomic concept $L = B$. Then, by (A.11), we have $\forall R.B \in J(\delta^v_{K_1 \sqsubseteq M_1})$; thus, there exists at least one clause $K' \sqsubseteq M' \sqcup \forall R.B \in S(v)$ with $\forall R.B \not\prec_v M'$ that produces $\forall R.B$ in $J(\delta^v_{K_1 \sqsubseteq M_1})$; hence $B \in \mathbf{B}_p$. But then $L \sqsubseteq L \mathrel{\hat{\in}} \mathcal{S}(u)$ holds by (A.13), which is stronger than the required $A \sqcap K_2 \sqsubseteq L \mathrel{\hat{\in}} \mathcal{S}(u)$. The case when $L = A$ follows directly from (A.13).
- Assume for contradiction that $A \sqcap K_2 \sqsubseteq M_2 \mathrel{\hat{\in}} \mathcal{S}(u)$. By Lemma 69, set $\mathcal{S}(v)$ is closed under $\prec_v$-hyperresolution with clause

  $$\exists R.A \sqcap \bigsqcap \{\forall R.B \mid \forall R.B \in J(\delta^v_{K_1 \sqsubseteq M_1})\} \sqsubseteq \bigsqcup \{C \mid \forall \mathsf{inv}(R).C \in \mathbf{L} \text{ and } C \notin J(\delta^v_{K_1 \sqsubseteq M_1})\}.$$

  Recall that $\exists R.A \in J(\delta^v_{K_1 \sqsubseteq M_1})$, so the above clause is clearly refuted in $J(\delta^v_{K_1 \sqsubseteq M_1})$. However, by Lemma 67, the clause is satisfied in $J(\delta^v_{K_1 \sqsubseteq M_1})$, which is a contradiction. Consequently, we have $A \sqcap K_2 \sqsubseteq M_2 \mathrel{\hat{\notin}} \mathcal{S}(u)$, as required.

Thus, we have $u \nvdash A \sqcap K_2 \sqsubseteq M_2$, which implies $\langle \delta^v_{K_1 \sqsubseteq M_1}, \delta^u_{A \sqcap K_2 \sqsubseteq M_2}, R \rangle \in E$ by the definition of E. Finally, $A \in J(\delta^u_{A \sqcap K_2 \sqsubseteq M_2})$ holds by Lemma 66. All of these observations prove that I satisfies property $(I_\exists)$ of Definition 59.

(Property $I_\forall$) Consider an arbitrary edge $\langle \delta^v_{K_1 \sqsubseteq M_1}, \delta^u_{A \sqcap K_2 \sqsubseteq M_2}, R \rangle \in E$ as in the definition of E, as well as arbitrary literals $\forall R.B$ and $\forall \mathsf{inv}(R).C$.

- Assume that $\forall R.B \in J(\delta^v_{K_1 \sqsubseteq M_1})$. Then $B \in K_2$ holds by (A.11); but then, Lemma 66 implies $J(\delta^u_{A \sqcap K_2 \sqsubseteq M_2}) \not\models A \sqcap K_2 \sqsubseteq M_2$. Consequently, we have $B \in J(\delta^u_{A \sqcap K_2 \sqsubseteq M_2})$, as required.
- Assume that $\forall \mathsf{inv}(R).C \in J(\delta^u_{A \sqcap K_2 \sqsubseteq M_2})$. Then $\forall \mathsf{inv}(R).C \in \mathbf{L}$ holds since $J(\delta^u_{A \sqcap K_2 \sqsubseteq M_2})$ is a literal interpretation of $\mathbf{L}$. Now, if $C \notin J(\delta^v_{K_1 \sqsubseteq M_1})$, then $\forall \mathsf{inv}(R).C \in M_2$ by (A.12), so $\forall \mathsf{inv}(R).C \notin J(\delta^u_{A \sqcap K_2 \sqsubseteq M_2})$ by Lemma 66, which contradicts our assumption. Consequently, we have $C \in J(\delta^v_{K_1 \sqsubseteq M_1})$, as required. □

Claim 72. *Pre-interpretation I is a pre-model of $\mathcal{O}$.*

Proof. Consider an arbitrary element $\delta^v_{K \sqsubseteq M} \in \Delta$ and an arbitrary normal clause $\alpha \in \mathcal{O}$. By Lemma 68, set $\mathcal{S}(v)$ is closed under $\prec_v$-hyperresolution with α, so $J(\delta^v_{K \sqsubseteq M}) \models \alpha$ by Lemma 67. This holds for arbitrary $\delta^v_{K \sqsubseteq M} \in \Delta$, so we have $I \models \alpha$. Finally, the latter holds for arbitrary $\alpha \in \mathcal{O}$, so I is a pre-model of $\mathcal{O}$. □

Claim 73. *Pre-interpretation I refutes each query $K \sqsubseteq M$ for which there exists a context $v \in \mathcal{V}$ such that v is complete for $K \sqsubseteq M$, $K \sqsubseteq L \mathrel{\hat{\in}} \mathcal{S}(v)$ for each $L \in K$, and $K \sqsubseteq M \mathrel{\hat{\notin}} \mathcal{S}(v)$.*

Proof. Let $K \sqsubseteq M$ be an arbitrary query satisfying the preconditions of the claim, and let $M' = M \cap \mathbf{L}$. Then $v \nvdash K \sqsubseteq M'$ holds as well, and this clause is over $\mathbf{L}$ by Claim 70. But then, $J(\delta^v_{K \sqsubseteq M'}) \not\models K \sqsubseteq M'$ by Lemma 66. Finally, each literal from $M \setminus M'$ is not from $\mathbf{L}$ and so it cannot occur in $J(\delta^v_{K \sqsubseteq M'})$; thus, $J(\delta^v_{K \sqsubseteq M'}) \not\models K \sqsubseteq M$ holds as well, so I refutes $K \sqsubseteq M$, as required. □

Appendix B. Proof of Theorem 37

We proceed similarly as in the proof of Theorem 6 in Appendix A. In the rest of Appendix B, we fix a normalized $\mathcal{ALCI}$ ontology $\mathcal{O}$, an admissible decomposition $\mathcal{D} = \langle \mathcal{V}, \mathcal{E}, \mathsf{core}, \mathsf{knw}, \mathsf{poss}, \prec, \vartheta \rangle$ of $\mathcal{O}$ and an arbitrary finite set of queries, and a clause system $\mathcal{S}$ for $\mathcal{D}$ satisfying the preconditions of Theorem 37; furthermore, we fix $\mathbf{L}$ to be the set of all literals that occur in $\mathcal{O}$, $\mathcal{D}$, or $\mathcal{S}$. We prove Theorem 37 by constructing a pre-interpretation that satisfies $\mathcal{O}$ but refutes the relevant queries. To simplify the presentation, given an ϵ-component $\mathcal{W}$ of $\mathcal{D}$, a context $v \in \mathcal{W}$, and a clause $K \sqsubseteq M$, we write $v \nvdash K \sqsubseteq M$ if and only if

- $K \subseteq \mathsf{poss}(v)$, $\;M \sqcap \mathsf{poss}(\mathcal{W}) \subseteq \mathsf{poss}(v)$, $\;$ and M is $\prec_v$-minimal (i.e., v is complete for $K \sqsubseteq M$),
- $K \sqsubseteq L \mathrel{\hat{\in}} \mathcal{S}(v)$ for each literal $L \in K$, and
- $K \sqsubseteq M \mathrel{\hat{\notin}} \mathcal{S}(v)$.

Theorem 37 holds vacuously if no context $v \in \mathcal{V}$ and no query $K \sqsubseteq M$ exist such that $v \nvdash K \sqsubseteq M$. Thus, we assume that $v \nvdash K \sqsubseteq M$ holds for at least one context $v \in \mathcal{V}$ and query $K \sqsubseteq M$; by the definition of clause strengthening, we then also have $v \nvdash K \sqsubseteq M'$ for $M' = M \cap \mathbf{L}$; the latter clause is over $\mathbf{L}$ by Claim 70. Let $I = \langle \Delta, E, J \rangle$ be defined as follows.

- Set Δ is the smallest set that contains a distinguished element $\delta^v_{K \sqsubseteq M}$ for each context $v \in \mathcal{V}$ and each clause $K \sqsubseteq M$ over $\mathbf{L}$ such that $v \nvdash K \sqsubseteq M$. Set Δ is not empty due to the above assumption.
- The value of function J is defined on each element $\delta^v_{K \sqsubseteq M} \in \Delta$ as follows. Let $\mathcal{W}$ be the ϵ-component of $\mathcal{D}$ that contains v, and let $w_0, w_1, \ldots, w_n$ be an ordering of the elements of $\mathcal{W}$ obtained by an arbitrary depth-first traversal of $\mathcal{D}_{\mathcal{W}}$ starting from $w_0 = v$. Conjunctions $K^0, K^1, \ldots, K^n$, disjunctions $M^0, M^1, \ldots, M^n$, and literal interpretations $J^0, J^1, \ldots, J^n$ are defined inductively as follows.
  - Case $i = 0$. Let $K^0 := K$, let $M^0 := M$, and let J^0 be an arbitrary literal interpretation of $\mathbf{L}$ w.r.t. $\mathcal{S}(w_0)$ and $\prec_{w_0}$ refuting $K^0 \sqsubseteq M^0$.
  - Case $i > 0$. Let w_j be the parent of w_i in the depth-first traversal of $\mathcal{D}_{\mathcal{W}}$; since w_j is considered before w_i in the depth-first traversal, K^j, M^j, and J^j have been defined. Let K^i and M^i be as follows:

    $$K^i := \bigsqcap \{ C \in \mathsf{poss}(w_j) \cap \mathsf{poss}(w_i) \mid C \in J^j \} \tag{B.1}$$

    $$M^i := \bigsqcup \{ C \in \mathsf{poss}(w_j) \cap \mathsf{poss}(w_i) \mid C \notin J^j \} \tag{B.2}$$

    Furthermore, let J^i be an arbitrary literal interpretation of $\mathbf{L}$ w.r.t. $\mathcal{S}(w_i)$ and $\prec_{w_i}$ refuting $K^i \sqsubseteq M^i$.

  Based on these definitions for $J^0, J^1, \ldots, J^n$, the value of function J on element $\delta^v_{K \sqsubseteq M}$ is defined by

  $$J(\delta^v_{K \sqsubseteq M}) := \mathsf{poss}(\mathcal{W}) \cap \left(\bigcup_{i=0}^{n} J^i \right). \tag{B.3}$$

- Set E is the smallest set containing $\langle \delta^v_{K_1 \sqsubseteq M_1}, \delta^u_{A \sqcap K_2 \sqsubseteq M_2}, R \rangle$ for all ϵ-components $\mathcal{W}$ and $\mathcal{U}$ of $\mathcal{D}$, each element $\delta^v_{K_1 \sqsubseteq M_1} \in \Delta$ with $v \in \mathcal{W}$, each literal $\exists R.A \in J(\delta^v_{K_1 \sqsubseteq M_1})$, and each edge $\langle v, u, \exists R.A \rangle \in \mathcal{E}$ with $u \in \mathcal{U}$ such that $u \nvdash A \sqcap K_2 \sqsubseteq M_2$, where K_2 and M_2 are as specified below.

  $$K_2 = \bigsqcap \{ B \mid \forall R.B \in J(\delta^v_{K_1 \sqsubseteq M_1}) \} \tag{B.4}$$

  $$M_2 = \bigsqcup \{ \forall \mathsf{inv}(R).C \in \mathsf{poss}(\mathcal{U}) \mid C \notin J(\delta^v_{K_1 \sqsubseteq M_1}) \} \tag{B.5}$$

In the last item, we have $K_2 \cup M_2 \subseteq \mathbf{L}$ and $A \in \mathbf{L}$; thus, $u \nvdash A \sqcap K_2 \sqsubseteq M_2$ ensures $\delta^u_{A \sqcap K_2 \sqsubseteq M_2} \in \Delta$, and so E is correctly defined. In contrast, the definition of $J(\delta^v_{K \sqsubseteq M})$ relies on the existence of literal interpretations that refute certain clauses, and it is not obvious that these literal interpretations exist; hence, the following lemma proves that this is the case.

Claim 74. *The value of $J(\delta^v_{K \sqsubseteq M})$ is correctly defined for each element $\delta^v_{K \sqsubseteq M} \in \Delta$.*

Proof. Consider an arbitrary element $\delta^v_{K \sqsubseteq M}$, and let $\mathcal{W}$, $w_0, w_1, \ldots, w_n$, $K^0, K^1, \ldots, K^n$, $M^0, M^1, \ldots, M^n$, and $J^0, J^1, \ldots, J^n$ be as specified above. We next inductively prove that $w_i \nvdash K^i \sqsubseteq M^i$ for each $0 \leq i \leq n$; this is sufficient to guarantee existence of the corresponding literal interpretation J^i. For $i = 0$, we have $w_0 \nvdash K^0 \sqsubseteq M^0$ immediately from the facts that $K^0 = K$, $M^0 = M$, and $\delta^v_{K \sqsubseteq M} \in \Delta$. Consider now an arbitrary integer $i > 0$, and let w_j be the parent of w_i in the depth-first traversal of $\mathcal{D}_{\mathcal{W}}$. Since the ϵ-edges of $\mathcal{D}$ are symmetric by admissibility condition (E1), the presence of the undirected edge $\{w_j, w_i\}$ in $\mathcal{D}_{\mathcal{W}}$ implies that $\langle w_j, w_i, \epsilon \rangle \in \mathcal{E}$. Now $w_i \nvdash K^i \sqsubseteq M^i$ follows from the following observations.

- Condition $K^i \subseteq \mathsf{poss}(w_i)$ holds trivially by the definition of K^i.

- Condition $M^i \cap \mathsf{poss}(\mathcal{W}) \subseteq \mathsf{poss}(w_i)$ holds trivially since $M^i \subseteq \mathsf{poss}(w_i)$ by the definition of M^i.
- Disjunction M^i is $\prec_{w_i}$-minimal by admissibility condition (P2) since $\langle w_j, w_i, \epsilon \rangle \in \mathcal{E}$ and $M^i \subseteq \mathsf{poss}(w_j) \cap \mathsf{poss}(w_i)$.
- Consider an arbitrary literal $L \in K^i$; we show $K^i \sqsubseteq L \mathrel{\hat{\in}} \mathcal{S}(w_i)$. By the definition of K^i, we have $L \in \mathsf{poss}(w_j) \cap \mathsf{poss}(w_i)$; hence, $\langle w_j, w_i, \epsilon \rangle \in \mathcal{E}$ implies $L \sqsubseteq L \mathrel{\hat{\in}} \mathcal{S}(w_i)$ by condition (I4) of Theorem 37, which is stronger than $K^i \sqsubseteq L \mathrel{\hat{\in}} \mathcal{S}(w_i)$.
- Assume for contradiction that $K^i \sqsubseteq M^i \mathrel{\hat{\in}} \mathcal{S}(w_i)$. Since $\langle w_j, w_i, \epsilon \rangle \in \mathcal{E}$ and $\mathcal{S}$ is closed under the Epsilon rule, $\mathcal{S}(w_j)$ is closed under $\prec_{w_j}$-hyperresolution with some strengthening of $K^i \sqsubseteq M^i$. By Lemmas 63 and 67, we then have $J^j \models K^i \sqsubseteq M^i$; however, by the definition K^i and M^i we clearly have $J^j \not\models K^i \sqsubseteq M^i$. Thus, $K^i \sqsubseteq M^i \mathrel{\hat{\notin}} \mathcal{S}(w_i)$ holds, as required. □

Next, we prove several important properties of J.

Claim 75. *Let $\delta^v_{K \sqsubseteq M} \in \Delta$ be an arbitrary element, and let $\mathcal{W}$ be the ϵ-component of $\mathcal{D}$ such that $v \in \mathcal{W}$. Then,*

1. $\mathsf{knw}(\mathcal{W}) \subseteq J(\delta^v_{K \sqsubseteq M}) \subseteq \mathsf{poss}(\mathcal{W})$,
2. *$J(\delta^v_{K \sqsubseteq M}) \not\models K \sqsubseteq M$, and*
3. *$J(\delta^v_{K \sqsubseteq M}) \models K' \sqsubseteq M'$ for each clause $K' \sqsubseteq M'$ for which a context $w \in \mathcal{W}$ exists such that $K' \cup M' \subseteq \mathsf{poss}(w)$ and $\mathcal{S}(w)$ is closed under $\prec_w$-hyperresolution with $K' \sqsubseteq M'$.*

Proof. Consider an arbitrary element $\delta^v_{K \sqsubseteq M} \in \Delta$; clearly, we have $v \not\vdash K \sqsubseteq M$. Furthermore, let $\mathcal{W}$ be the ϵ-component of $\mathcal{D}$ such that $v \in \mathcal{W}$, and let $w_0, w_1, \ldots, w_n$, $K^0, K^1, \ldots, K^n$, $M^0, M^1, \ldots, M^n$, and $J^0, J^1, \ldots, J^n$ be as specified in the definition of the pre-interpretation I. Before proceeding, we first prove the following auxiliary property ($*$).

For all integers $0 \leq i, j \leq n$ and each literal $L \in \mathsf{poss}(w_i) \cap \mathsf{poss}(w_j)$, we have $L \in J^i$ if and only if $L \in J^j$.

Since $\mathcal{D}_{\mathcal{W}}$ is a tree by admissibility condition (E2), the shortest path between w_i and w_j is unique; we now prove ($*$) by induction on the length of this path. Towards this goal, consider an arbitrary literal $L \in \mathsf{poss}(w_i) \cap \mathsf{poss}(w_j)$; we have the following cases.

- If $i = j$, then $J^i = J^j$, so clearly $L \in J^i$ if and only if $L \in J^j$.
- Assume that w_j is the parent of w_i in the traversal of $\mathcal{D}_{\mathcal{W}}$. Then $K^i \cup M^i = \mathsf{poss}(w_i) \cap \mathsf{poss}(w_j)$ by the definition of K^i and M^i, which implies $L \in K^i \cup M^i$. Furthermore, by Lemma 66 we have $J^i \not\models K^i \sqsubseteq M^i$. But then, if $L \in K^i$, then by the definition of K^i we have $L \in J^j$, and $J^i \not\models K^i \sqsubseteq M^i$ implies $L \in J^i$. Analogously, if $L \in M^i$, then by the definition of M^i we have $L \notin J^j$, and $J^i \not\models K^i \sqsubseteq M^i$ implies $L \notin J^i$.
- The case when w_i is the parent of w_j is symmetric to the previous one.
- The remaining possibility is that the length of the shortest path between w_i and w_j is greater than one, so let w_k be an arbitrary vertex on this path different from w_i and w_j. By admissibility condition (E3), we have $L \in \mathsf{poss}(w_k)$. Furthermore, the paths between w_i and w_k, and between w_k and w_j are both shorter than the path between w_i and w_j, so property ($*$) holds for i and k, and for k and j. But then, property ($*$) clearly holds for i and j as well.

Property ($*$) clearly implies that, for each integer $0 \leq i \leq n$ and each literal $L \in \mathsf{poss}(w_i)$, we have $L \in J(\delta^v_{K \sqsubseteq M})$ if and only if $L \in J^i$; we call this property (†). We are now ready to prove Properties 1–3 of this claim.

(Property 1) The definition of $J(\delta^v_{K \sqsubseteq M})$ in (B.3) clearly implies $J(\delta^v_{K \sqsubseteq M}) \subseteq \mathsf{poss}(\mathcal{W})$. In order to prove $\mathsf{knw}(\mathcal{W}) \subseteq J(\delta^v_{K \sqsubseteq M})$, consider an arbitrary literal $L \in \mathsf{knw}(\mathcal{W})$; clearly, a context $w_i \in \mathcal{W}$ exists such that $L \in \mathsf{knw}(w_i)$. By condition (I2) of Theorem 37, we have $\top \sqsubseteq L \mathrel{\hat{\in}} \mathcal{S}(w_i)$; hence $L \in J^i$ by Lemma 65. Consequently, $L \in J(\delta^v_{K \sqsubseteq M})$ holds by property (†), as required.

(Property 2) By Lemma 66, $K^0 = K$, and $M^0 = M$, we have $J^0 \not\models K \sqsubseteq M$—that is, $K \subseteq J^0$ and $M \cap J^0 = \emptyset$. Now $v \not\vdash K \sqsubseteq M$ implies $K \subseteq \mathsf{poss}(v)$ and $M \cap \mathsf{poss}(\mathcal{W}) \subseteq \mathsf{poss}(v)$, so property (†) implies $K \subseteq J(\delta^v_{K \sqsubseteq M})$ and $[M \cap \mathsf{poss}(\mathcal{W})] \cap J(\delta^v_{K \sqsubseteq M}) = \emptyset$. Furthermore, since $\mathsf{poss}(\mathcal{W}) \cap J(\delta^v_{K \sqsubseteq M}) = J(\delta^v_{K \sqsubseteq M})$, we have $M \cap J(\delta^v_{K \sqsubseteq M}) = \emptyset$; thus, $J(\delta^v_{K \sqsubseteq M}) \not\models K \sqsubseteq M$ holds, as required.

(Property 3) Let $K' \sqsubseteq M'$ be a clause and let $w_i \in \mathcal{W}$ be a context such that $K' \cup M' \subseteq \mathsf{poss}(w_i)$ and set $\mathcal{S}(w_i)$ is closed under $\prec_{w_i}$-hyperresolution with $K' \sqsubseteq M'$. By Lemma 67, we have $J^i \models K' \sqsubseteq M'$. But then, since $K' \cup M' \subseteq \mathsf{poss}(w_i)$, property (†) implies $J(\delta^v_{K \sqsubseteq M}) \models K' \sqsubseteq M'$, as required. □

To complete the proof of Theorem 37, the following claims show that structure I is a pre-interpretation, and that it satisfies $\mathcal{O}$ but refutes the relevant queries.

Claim 76. *Structure I satisfies conditions ($I_\exists$) and ($I_\forall$) of pre-interpretations from Definition 59.*

Proof. (Condition $I_\exists$) Consider an arbitrary element $\delta^v_{K_1 \sqsubseteq M_1} \in \Delta$ and an arbitrary literal $\exists R.A \in J(\delta^v_{K_1 \sqsubseteq M_1})$; we show that an edge $\langle \delta^v_{K_1 \sqsubseteq M_1}, \gamma, R \rangle \in E$ exists such that $A \in J(\gamma)$. Let $\mathcal{W}$ be the ϵ-component of $\mathcal{D}$ such that $v \in \mathcal{W}$. By admissibility condition (S3), an edge $\langle w, u, \exists R.A \rangle \in \mathcal{E}$ exists such that $w \in \mathcal{W}$ and conditions (B.6) and (B.7) are both satisfied.

$$\mathsf{poss}(u) \supseteq \{A\} \cup \{B \mid \forall R.B \in \mathsf{poss}(\mathcal{W})\} \tag{B.6}$$

$$\mathsf{poss}(w) \supseteq \{\exists R.A\} \cup \{\forall R.B \mid \forall R.B \in \mathsf{poss}(\mathcal{W})\} \tag{B.7}$$

Let $\mathcal{U}$ be the ϵ-component of $\mathcal{D}$ such that $u \in \mathcal{U}$, and let K_2 and M_2 be as specified in (B.4) and (B.5), respectively. The following observations prove that $u \not\vdash A \sqcap K_2 \sqsubseteq M_2$ holds.

- Condition $A \sqcap K_2 \subseteq \mathsf{poss}(u)$ follows from (B.4), Property 1 of Claim 75, and (B.6).
- Admissibility condition (S2) implies $M_2 \cap \mathsf{poss}(\mathcal{U}) \subseteq \mathsf{poss}(u)$.
- Disjunction M_2 is $\prec_u$-minimal since, by admissibility condition (P1), $\langle w, u, \exists R.A \rangle \in \mathcal{E}$ implies that $\prec_u$ is R-admissible.
- Consider an arbitrary literal $L \in A \sqcap K_2$; we show that $A \sqcap K_2 \sqsubseteq L \mathrel{\hat{\in}} \mathcal{S}(u)$. Assume that $L \in K_2$, so L is an atomic concept $L = B$. Then, we have $\forall R.B \in J(\delta^v_{K_1 \sqsubseteq M_1}) \subseteq \mathsf{poss}(\mathcal{W})$ by (B.4) and Property 1 of Claim 75. Thus, we have $B \in \mathsf{poss}(u)$ and $\forall R.B \in \mathsf{poss}(w)$ by (B.6) and (B.7), and then $B \sqsubseteq B \mathrel{\hat{\in}} \mathcal{S}(u)$ by condition (I3) of Theorem 37, which is stronger than the required $A \sqcap K_2 \sqsubseteq B \mathrel{\hat{\in}} \mathcal{S}(u)$. The proof for the case $L = A$ is analogous.
- Assume for contradiction that $A \sqcap K_2 \sqsubseteq M_2 \mathrel{\hat{\in}} \mathcal{S}(u)$. By admissibility condition (S2), we have $C \in \mathsf{poss}(w)$ for each $\forall \mathsf{inv}(R).C \in M_2$. Thus, since the Pred rule from Table 5 is not applicable to $\mathcal{S}(v)$, the Pred from Table 3 is not applicable to $\mathcal{S}(v)$ either. Hence, by Lemma 69, set $\mathcal{S}(w)$ is closed under $\prec_w$-hyperresolution with the clause

  $$\exists R.A \sqcap \bigsqcap \{\forall R.B \mid \forall R.B \in J(\delta^v_{K_1 \sqsubseteq M_1})\} \sqsubseteq \bigsqcup \{C \mid \forall \mathsf{inv}(R).C \in \mathsf{poss}(\mathcal{U}) \text{ and } C \notin J(\delta^v_{K_1 \sqsubseteq M_1})\}.$$

  Recall that $\exists R.A \in J(\delta^v_{K_1 \sqsubseteq M_1})$, so the above clause is clearly refuted in $J(\delta^v_{K_1 \sqsubseteq M_1})$. However, all literals on the left-hand side of this clause are contained in $\mathsf{poss}(w)$ by (B.7), and each concept C on its right-hand side is contained in $\mathsf{poss}(w)$; but then, Property 3 of Claim 75 implies that this clause is satisfied in $J(\delta^v_{K_1 \sqsubseteq M_1})$, which is a contradiction. Consequently, we have $A \sqcap K_2 \sqsubseteq M_2 \mathrel{\hat{\notin}} \mathcal{S}(u)$, as required.

Thus, we have $u \not\vdash A \sqcap K_2 \sqsubseteq M_2$, which implies $\langle \delta^v_{K_1 \sqsubseteq M_1}, \delta^u_{A \sqcap K_2 \sqsubseteq M_2}, R \rangle \in E$ by the definition of E. Finally, $A \in J(\delta^u_{A \sqcap K_2 \sqsubseteq M_2})$ holds by Property 2 of Claim 75. This concludes the proof of condition $(I_\exists)$.

(Condition $I_\forall$). Consider an arbitrary edge $\langle \delta^v_{K_1 \sqsubseteq M_1}, \delta^u_{A \sqcap K_2 \sqsubseteq M_2}, R \rangle \in E$ and arbitrary literals $\forall R.B$ and $\forall \mathsf{inv}(R).C$, and let $\mathcal{U}$ be the ϵ-component of $\mathcal{D}$ such that $u \in \mathcal{U}$.

- Assume that $\forall R.B \in J(\delta^v_{K_1 \sqsubseteq M_1})$. Then $B \in K_2$ by (B.4); hence Property 2 of Claim 75 implies $J(\delta^u_{A \sqcap K_2 \sqsubseteq M_2}) \not\models A \sqcap K_2 \sqsubseteq M_2$; therefore $B \in J(\delta^u_{A \sqcap K_2 \sqsubseteq M_2})$, as required.
- Assume that $\forall \mathsf{inv}(R).C \in J(\delta^u_{A \sqcap K_2 \sqsubseteq M_2})$. Then $\forall \mathsf{inv}(R).C \in \mathsf{poss}(\mathcal{U})$ by Property 1 of Claim 75. Now, if $C \notin J(\delta^v_{K_1 \sqsubseteq M_1})$, then (B.5) implies $\forall \mathsf{inv}(R).C \in M_2$, so $\forall \mathsf{inv}(R).C \notin J(\delta^u_{A \sqcap K_2 \sqsubseteq M_2})$ follows from Property 2 of Claim 75, which contradicts the assumption. Thus, we have $C \in J(\delta^v_{K_1 \sqsubseteq M_1})$, as required. □

Claim 77. *Pre-interpretation I is a pre-model of $\mathcal{O}$.*

Proof. Consider an arbitrary element $\delta^v_{K \sqsubseteq M} \in \Delta$ and an arbitrary normal clause $K' \sqsubseteq M' \in \mathcal{O}$; we show that $J(\delta^v_{K \sqsubseteq M})$ satisfies $K' \sqsubseteq M'$. This is obvious if $K' \nsubseteq J(\delta^v_{K \sqsubseteq M})$, so assume that $K' \subseteq J(\delta^v_{K \sqsubseteq M})$. Let $\mathcal{W}$ be the ϵ-component of $\mathcal{D}$ such that $v \in \mathcal{W}$. Then, Property 1 of Claim 75 implies $K' \subseteq \mathsf{poss}(\mathcal{W})$. Furthermore, by the ontology condition of Definition 36, a context $w \in \mathcal{W}$ exists such that $K' \cup M' \subseteq \mathsf{poss}(w)$. Since $\mathcal{S}$ is closed under the Hyper rule from Table 5, set $\mathcal{S}(w)$ is closed under $\prec_w$-hyperresolution with $K' \sqsubseteq M'$. But then, Property 3 of Claim 75 implies $J(\delta^v_{K \sqsubseteq M}) \models K' \sqsubseteq M'$, as required. □

Claim 78. *Pre-interpretation I refutes each query $K \sqsubseteq M$ for which there exists a context $v \in \mathcal{V}$ such that v is complete for $K \sqsubseteq M$, $K \sqsubseteq L \mathrel{\hat{\in}} \mathcal{S}(v)$ for each $L \in K$, and $K \sqsubseteq M \mathrel{\hat{\notin}} \mathcal{S}(v)$.*

Proof. Let $K \sqsubseteq M$ be an arbitrary query satisfying the preconditions of the claim, and let $M' = M \cap \mathbf{L}$. Then $v \not\vdash K \sqsubseteq M'$ holds as well, and this clause is over $\mathbf{L}$ by Claim 70. But then, $J(\delta^v_{K \sqsubseteq M'}) \not\models K \sqsubseteq M'$ by Property 2 of Claim 75. Finally, each literal from $M \setminus M'$ is not from $\mathbf{L}$ so it cannot occur in $J(\delta^v_{K \sqsubseteq M'})$; thus, $J(\delta^v_{K \sqsubseteq M'}) \not\models K \sqsubseteq M$ holds as well, so I refutes $K \sqsubseteq M$, as required. □

References

[1] A. Sidhu, T. Dillon, E. Chang, B. S. Sidhu, Protein Ontology Development using OWL, in: Proc. of the OWL: Experiences and Directions Workshop (OWLED 2005), volume 188 of *CEUR WS Proceedings*, Galway, Ireland.

[2] C. Golbreich, S. Zhang, O. Bodenreider, The Foundational Model of Anatomy in OWL: Experience and Perspectives, Journal of Web Semantics 4 (2006) 181–195.

[3] J. Goodwin, Experiences of using OWL at the Ordnance Survey, in: Proc. of the OWL: Experiences and Directions Workshop (OWLED 2005), volume 188 of *CEUR WS Proceedings*, Galway, Ireland.

[4] S. Derriere, A. Richard, A. Preite-Martinez, An Ontology of Astronomical Object Types for the Virtual Observatory, in: Proc. of the 26th meeting of the IAU: Virtual Observatory in Action: New Science, New Technology, and Next Generation Facilities, Prague, Czech Republic, pp. 17–18.

[5] D. Soergel, B. Lauser, A. Liang, F. Fisseha, J. Keizer, S. Katz, Reengineering Thesauri for New Applications: The AGROVOC Example, Journal of Digital Information 4 (2004).

[6] L. Lacy, G. Aviles, K. Fraser, W. Gerber, A. Mulvehill, R. Gaskill, Experiences Using OWL in Military Applications, in: Proc. of the OWL: Experiences and Directions Workshop (OWLED 2005), volume 188 of *CEUR WS Proceedings*, Galway, Ireland.

[7] P. F. Patel-Schneider, P. Hayes, I. Horrocks, OWL Web Ontology Language: Semantics and Abstract Syntax, W3C Recommendation, `http://www.w3.org/TR/owl-semantics/`, 2004.

[8] B. Motik, P. F. Patel-Schneider, B. Parsia, OWL 2 Web Ontology Language: Structural Specification and Functional-Style Syntax, W3C Recommendation, `http://www.w3.org/TR/owl2-syntax/`, 2009.

[9] B. Cuenca Grau, I. Horrocks, B. Motik, B. Parsia, P. Patel-Schneider, U. Sattler, OWL 2: The next step for OWL, Journal of Web Semantics: Science, Services and Agents on the World Wide Web 6 (2008) 309–322.

[10] F. Baader, D. Calvanese, D. McGuinness, D. Nardi, P. F. Patel-Schneider (Eds.), The Description Logic Handbook: Theory, Implementation and Applications, Cambridge University Press, 2nd edition, 2007.

[11] H. Andréka, J. van Benthem, I. Németi, Modal Languages and Bounded Fragments of Predicate Logic, Journal of Philosophical Logic 27 (1998) 217–274.

[12] B. Parsia, E. Sirin, Pellet: An OWL-DL Reasoner, Poster at the 3rd Int. Semantic Web Conference (ISWC 2004), 2004.

[13] D. Tsarkov, I. Horrocks, FaCT++ Description Logic Reasoner: System Description, in: Proc. of the 3rd Int. Joint Conf. on Automated Reasoning (IJCAR 2006), volume 4130 of *LNAI*, Springer, Seattle, WA, USA, 2006, pp. 292–297.

[14] B. Motik, R. Shearer, I. Horrocks, Hypertableau Reasoning for Description Logics, Journal of Artificial Intelligence Research 36 (2009) 165–228.

[15] R. S. Gonçalves, B. Parsia, U. Sattler, Performance Heterogeneity and Approximate Reasoning in Description Logic Ontologies, in: P. Cudré-Mauroux, J. Heflin, E. Sirin, T. Tudorache, J. Euzenat, M. Hauswirth, J. X. Parreira, J. Hendler, G. Schreiber, A. Bernstein, E. Blomqvist (Eds.), Proc. of the 11th Int. Semantic Web Conference (ISWC 2012), volume 7649 of *LNCS*, Springer, Boston, MA, USA, 2012, pp. 82–98.

[16] U. Hustadt, B. Motik, U. Sattler, Data Complexity of Reasoning in Very Expressive Description Logics, in: Proc. of the 19th Int. Joint Conf. on Artificial Intelligence (IJCAI 2005), Morgan Kaufmann Publishers, Edinburgh, UK, 2005, pp. 466–471.

[17] M. Krötzsch, S. Rudolph, P. Hitzler, Complexity Boundaries for Horn Description Logics, in: Proc. of the 22nd National Conference on Artificial Intelligence (AAAI 2007), AAAI Press, Vancouver, BC, Canada, 2007, pp. 452–457.

[18] F. Baader, S. Brandt, C. Lutz, Pushing the $\mathcal{EL}$ Envelope, in: L. P. Kaelbling, A. Saffiotti (Eds.), Proc. of the 19th Int. Joint Conference on Artificial Intelligence (IJCAI 2005), Morgan Kaufmann Publishers, Edinburgh, UK, 2005, pp. 364–369.

[19] D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, R. Rosati, Tractable Reasoning and Efficient Query Answering in Description Logics: The DL-Lite Family, Journal of Automated Reasoning 9 (2007) 385–429.

[20] A. Artale, D. Calvanese, R. Kontchakov, M. Zakharyaschev, The DL-Lite Family and Relations, Journal of Artificial Intelligence Research 36 (2009) 1–69.

[21] Y. Kazakov, Consequence-Driven Reasoning for Horn SHIQ Ontologies, in: C. Boutilier (Ed.), Proc. of the 21st Int. Joint Conf. on Artificial Intelligence (IJCAI 2009), Pasadena, CA, USA, pp. 2040–2045.

[22] F. Simančík, Y. Kazakov, I. Horrocks, Consequence-Based Reasoning beyond Horn Ontologies, in: T. Walsh (Ed.), Proc. of the 22nd Int. Joint Conf. on Artificial Intelligence (IJCAI 2011), Barcelona, Spain, pp. 1093–1098.

[23] Y. Kazakov, M. Krötzsch, F. Simančík, Concurrent Classification of $\mathcal{EL}$ Ontologies, in: L. Aroyo, C. Welty, H. Alani, J. Taylor, A. Bernstein, L. Kagal, N. F. Noy, E. Blomqvist (Eds.), Proc. of the 10th Int. Semantic Web Conference (ISWC 2011), volume 7031 of *LNCS*, Springer, Boston, MA, USA, 2011, pp. 305–320.

[24] R. G. Downey, M. R. Fellows, Parameterized Complexity, Springer, 1999.

[25] N. Robertson, P. Seymour, Graph minors. III. Planar tree-width, Journal of Combinatorial Theory, Series B 36 (1984) 49–64.

[26] S. Szeider, On Fixed-Parameter Tractable Parameterizations of SAT, in: E. Giunchiglia, A. Tacchella (Eds.), Proc. of the 6th Int. Conf. on Theory and Applications of Satisfiability Testing (SAT 2003), volume 2919 of *LNCS*, Springer, Santa Margherita Ligure, Italy, 2003, pp. 188–202.

[27] C. Lutz, F. Wolter, Non-Uniform Data Complexity of Query Answering in Description Logics, in: G. Brewka, T. Eiter, S. A. McIlraith (Eds.), Proc. of the 13th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR 2012), AAAI Press, Rome, Italy, 2012, pp. 297–307.

[28] S. Demri, H. de Nivelle, Deciding Regular Grammar Logics with Converse Through First-Order Logic, Journal of Logic, Language and Information 14 (2005) 289–329.

[29] H. L. Bodlaender, A Linear-Time Algorithm for Finding Tree-Decompositions of Small Treewidth, SIAM Journal on Computing 25 (1996) 1305–1317.

[30] C. G. Fermüller, A. Leitsch, U. Hustadt, T. Tammet, Resolution Decision Procedures, in: A. Robinson, A. Voronkov (Eds.), Handbook of Automated Reasoning, volume II, Elsevier Science, 2001, pp. 1791–1849.

[31] L. Bachmair, H. Ganzinger, Resolution Theorem Proving, in: A. Robinson, A. Voronkov (Eds.), Handbook of Automated Reasoning, volume I, Elsevier Science, 2001, pp. 19–99.

[32] B. Motik, U. Sattler, A Comparison of Reasoning Techniques for Querying Large Description Logic ABoxes, in: M. Hermann, A. Voronkov (Eds.), Proc. of the 13th Int. Conf. on Logic for Programming Artificial Intelligence and Reasoning (LPAR 2006), volume 4246 of *LNCS*, Springer, Phnom Penh, Cambodia, 2006, pp. 227–241.

[33] R. Goré, L. A. Nguyen, EXPTIME Tableaux with Global Caching for Description Logics with Transitive Roles, Inverse Roles and Role Hierarchies, in: N. Olivetti (Ed.), Proc. of the 16th Int. Conf. on Automated Reasoning with Tableaux and Related Methods (TABLEAUX 2007), volume 4548 of *LNCS*, Springer, Aix en Provence, France, 2007, pp. 133–148.

[34] R. Goré, L. A. Nguyen, ExpTime Tableaux for $\mathcal{ALC}$ Using Sound Global Caching, Journal of Automated Reasoning 50 (2013) 355–381.

[35] C. Weidenbach, Combining superposition, sorts and splitting, in: A. Robinson, A. Voronkov (Eds.), Handbook of Automated Reasoning, volume II, Elsevier Science, 2001, pp. 1965–2013.

[36] S. Schulz, System Description: E 0.81, in: D. A. Basin, M. Rusinowitch (Eds.), Proc. of the 2nd Int. Conf. on Automated Reasoning (IJCAR 2004), volume 3097 of *LNCS*, Springer, Cork, Ireland, 2004, pp. 223–228.

[37] B. Glimm, I. Horrocks, B. Motik, G. Stoilos, A Novel Approach to Ontology Classification, Journal of Web Semantics: Science, Services and Agents on the World Wide Web 14 (2012) 84–101.