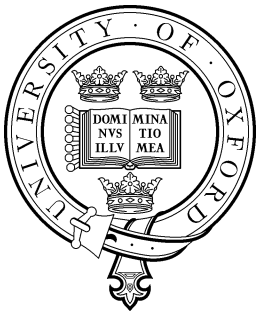
Consequence-Based Reasoning for $\mathcal{SRIQ}$ Ontologies



Andrew Bate
Pembroke College
University of Oxford

A thesis submitted for the degree of
Doctor of Philosophy
Trinity 2016

Abstract

Description logics (DLs) are knowledge representation formalisms with numerous applications and well-understood model-theoretic semantics and computational properties. $\mathcal{SRIQ}$ is a DL that provides the logical underpinning for the semantic web language OWL 2, which is the W3C standard for knowledge representation on the web.

A central component of most DL applications is an efficient and scalable reasoner, which provides services such as consistency testing and classification. Despite major advances in DL reasoning algorithms over the last decade, however, ontologies are still encountered in practice that cannot be handled by existing DL reasoners.

Consequence-based calculi are a family of reasoning techniques for DLs. Such calculi have proved very effective in practice and enjoy a number of desirable theoretical properties. Up to now, however, they were proposed for either Horn DLs (which do not support disjunctive reasoning), or for DLs without cardinality constraints. In this thesis we present a novel consequence-based algorithm for TBox reasoning in $\mathcal{SRIQ}$—a DL that supports both disjunctions and cardinality constraints. Combining the two features is non-trivial since the intermediate consequences that need to be derived during reasoning cannot be captured using DLs themselves. Furthermore, cardinality constraints require reasoning over equality, which we handle using the framework of *ordered paramodulation*—a state-of-the-art method for equational theorem proving. We thus obtain a calculus that can handle an expressive DL, while still enjoying all the favourable properties of existing consequence-based algorithms, namely optimal worst-case complexity, one-pass classification, and pay-as-you-go behaviour.

To evaluate the practicability of our calculus, we implemented it in *Sequoia*—a new DL reasoning system. Empirical results show substantial robustness improvements over well-established algorithms and implementations, and performance competitive with closely related work.

Acknowledgements

I would like to express my gratitude to my supervisors Bernardo Cuenca Grau, Boris Motik, and Ian Horrocks. To Bernardo, for teaching me how to present my work, and for consistently supporting me over the years. It has been greatly appreciated in times of struggle. To Boris, for dedicating the time to discuss ideas and helping me to improve my technical work. To Ian, for always finding the time for me when I needed advice.

I am also grateful to my colleagues and friends in the Knowledge Representation and Reasoning research group, in particular Yavor Nenov and Robert Piro, who have made the journey much easier.

This work was supported by a studentship from the Engineering and Physical Sciences Research Council (EPSRC).

Andrew Bate

Pembroke College, Oxford

Trinity 2016

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Knowledge representation and reasoning (KRR) is the area of artificial intelligence concerned with how human expert knowledge can be represented symbolically and manipulated in an automated way by reasoning programs [vHLP07]. Two key areas of KRR research are *knowledge representation formalisms* and *reasoning algorithms*.

Knowledge Representation Formalisms An *ontology* is an explicit specification of a body of knowledge in a particular domain of interest, and consists of the definition of entities and the relationships that hold amongst them [Gru93]. An ontology language is a formal language used to construct ontologies. Research into knowledge representation formalisms aims to establish ontology languages that allow for the representation of knowledge in a manner that is unambiguous, that provides enough expressive power for realistic applications and that can be efficiently processed by machines. Numerous formalisms have been proposed to meet this goal, with origins in both academic and commercial environments. The Web Ontology Language (OWL) [CHMP+08], a recommendation of the World Wide Web Consortium, is a family of ontology languages. Ontologies are intrinsic to the Semantic Web and are used in many application areas including biomedical information systems [SDCS05, GZB06, RR06, ABBB+00], medicine [Spa00, RM03], biology [SARB+07], agriculture [SLLF+04], astronomy [DRP07], defence [LAFG+05], geography [Goo05], and aerospace [HG08].

Reasoning Algorithms Ontology languages are equipped with logic-based semantics, and so allow us to infer logical consequences of the knowledge explicitly stated in the ontology. This

process can be automated using a software component called a *reasoner*. Reasoner support is essential for authoring ontologies and using them in practice [Hor11]. For example, a reasoner can be used to detect inconsistencies, which may indicate a modelling error. Furthermore, ontologies are commonly used together with data, and a reasoner supports querying implicit information logically entailed by an ontology and the data. In particular, *ontology-based data access* uses an ontology to provide a unified, conceptual view of data that may be heterogeneous and stored in multiple sources [CCKK+15]. Ontology editors serve as an interface between a user and an ontology, and use a reasoner to facilitate tasks such as visualisation of knowledge, query answering and detection of inconsistencies.

Description Logics (DLs) [BCMN+10] are a family of logic-based knowledge representation languages that provide the theoretical underpinning for OWL 2 [MCHW+09]. A DL *ontology* allows for the modelling of a domain of interest using the following key notions: *concepts* (which represent sets of objects in the domain), *roles* (which represent binary relations between objects in the domain), and *individual names* (which represent single objects in the domain). In the terminology of first-order logic, these are unary predicates, binary predicates, and constants, respectively. Concepts and roles can either be atomic, which means that they are named only, or complex, which means that they are defined in terms of other concepts and roles using constructors. Complex concept descriptions can be built using concept constructors. Different DLs differ in which constructors they support. Commonly occurring constructors are the Boolean operators and existential and universal restrictions. For example, the complex concept

$$\mathsf{Director} \sqcap \exists \mathsf{hasDirected}.\,(\mathsf{Film} \sqcap \exists \mathsf{wonAward}.\mathsf{AcademyAward})$$

represents the set of directors who have each won an Academy Award for at least one of their films. In order to facilitate the design of practical reasoning algorithms, DLs only provide a subset of the expressivity of first-order logic [BCMN+10, Bor96]. Most DLs are strongly related to either the *guarded fragment* [ANvB98] or the *two-variable fragment with counting* ($\mathcal{C}^2$) [GOR97]. Nonetheless, the use of DLs in practice has shown that they provide epistemologically adequate constructors for modelling complex domains [Spa00].

Subsumption is the problem of determining whether the set denoted by a concept C is a subset of the set denoted by a concept D in all models of an ontology [BCMN+10, Chapter 2], and it

is a fundamental reasoning problem in applications of DLs. The preorder of concept subsumptions is known as the *taxonomy*. For users, this is often the most convenient form of viewing the content of an ontology. For ontology engineers, examining whether the subsumption hierarchy conforms to their expectations is a good way of testing the correctness of their modelling. Classification, the task of computing the taxonomy, is therefore one of the most important ontology reasoning tasks. For expressive DLs, this problem is of high worst-case complexity,[1] ranging from EXPTIME-complete up to 2NEXPTIME-complete [Kaz08].

Despite these complexity bounds, highly optimised reasoners such as FaCT++ [TH06], Pellet [SPCK+07], Racer [HM01], HermiT [MSH09], and Konclude [SLG14] have proved successful in practice. These systems are typically based on (hyper)tableau calculi that, for a given ontology, construct a finite representation of a canonical model disproving a postulated subsumption. While such calculi can handle many ontologies, in some cases they construct very large model representations, which is a source of performance problems; this is further exacerbated by the large number of subsumption tests often required to classify an ontology.

1.1 Consequence-Based Reasoning

A breakthrough in ontology reasoning came in the form of *consequence-based calculi*. Such calculi have the following desirable features.

One-Pass Classification Consequence-based calculi are not only refutation complete, but can also (dis)prove all relevant subsumptions in a single run, known as *one-pass classification*, which can greatly reduce the overall computational work during classification.

Worst-Case Optimal Unlike implemented (hyper)tableau reasoners, consequence-based calculi are worst-case optimal for the logic they support.

Pay-As-You-Go (PAYG) Behaviour For a procedure to qualify as having PAYG behaviour, at a minimum it must be worst-case optimal for sublogics. For example, a PAYG algorithm for classification of a $\mathcal{SHIQ}$ ontology must exhibit worst-case PTIME complexity for the classification of an $\mathcal{ELH}$ ontology. Furthermore, the procedure should be insensitive to syntactic changes

[1] In this thesis, all complexity results cited from the literature for the fundamental DL inference problems assume that the numbers used in cardinality constraints are represented in unary.

to the ontology that do not change the interactions between axioms in the input. For example, rewriting the axiom $\exists R.A \sqsubseteq B$ as $A \sqsubseteq \forall R^{-}.B$ should not change the behaviour of the algorithm. Finally, if an ontology that falls within a sublogic has a more expressive axiom added to it, then inferences that do not involve this new axiom should be unaffected by this addition. For example, a procedure for classification would not be considered PAYG if it were to use a specialised set of inference rules for $\mathcal{ELH}$ ontologies, and switch to a different set of inference rules for ontologies that do not fall within the $\mathcal{ELH}$ fragment.

The reasoning algorithm by Baader et al. [BBL05] for the lightweight logic $\mathcal{EL}^{++}$ can be considered to be the first consequence-based calculus for DLs. Consequence-based calculi were later developed for the more expressive Horn-$\mathcal{SHIQ}$ [Kaz09] and Horn-$\mathcal{SROIQ}$ [ORS10]—DLs that support (a restricted form of) counting quantifiers, but not concept disjunction. Consequence-based calculi were also developed for $\mathcal{ALCH}$ [SKH11] and $\mathcal{ALCI}$ [SMH14], which support concept disjunction, but not counting quantifiers.

Consequence-based reasoners for $\mathcal{EL}$ include ELK [KKS11], Snorocket [LB10], CEL [BLS06], and jcel [Men12], and prototype implementations for more expressive logics exist and preliminary evaluation has shown their potential to perform well in practice [Kaz09, SKH11]. The consequence-based reasoner CB for Horn-$\mathcal{SHIF}$ was the first to classify a large version of GALEN [Kaz09]. However, an extensive evaluation for consequence-based procedures beyond $\mathcal{EL}$ has not previously been conducted.

1.2 Open Problems

Existing consequence-based algorithms can handle either disjunctions or cardinality constraints, but not both. As we discuss in detail in Chapter 4, it is challenging to extend these existing algorithms to DLs such as $\mathcal{ALCHIQ}^{+}$ or $\mathcal{SRIQ}$ that combine both kinds of constructs: cardinality constraints require equality reasoning, which together with disjunctions, can impose complex constraints on ontology models. Furthermore, unlike in existing consequence-based calculi, such constraints cannot be captured in the $\mathcal{SRIQ}$ logic: due to the interplay between inverse roles and cardinality constraints, one may need to consider equality between a successor and the predecessor of an element (in a tree-shaped model). This makes the reasoning algorithm much more involved because these non-trivial constraints will have to be captured either by inference

rules that derive all required consequences of such constraints, or alternatively, the constraints need to be represented in an adequately expressive logic. Currently, there is no known consequence-based calculus that can handle a DL that allows for both disjunctions and cardinality constraints.

Steigmiller and Glimm [SG15] presented a way of combining a consequence-based calculus with a traditional tableau-based prover for the DL $\mathcal{SROIQ}$. However, while such a combination seems to perform well in practice, the saturation rules are only known to be complete for $\mathcal{ELH}$ ontologies, and the overall approach is not worst-case optimal for more expressive DLs such as $\mathcal{ALCHIQ}$ [HKS05].

Other techniques have been proposed to combine reasoning algorithms. For instance, the reasoning system MORe [ACH12] uses module extraction to identify a subset of an ontology for which classification is complete using a specialised algorithm (such as a consequence-based procedure for $\mathcal{EL}$), and only uses a fully-fledged algorithm (such as a tableau-based procedure) for the remaining part of the ontology. However, such an approach does not allow for one-pass classification (unless the entire ontology can be handled using the specialised algorithm), nor is the combined approach worst-case optimal.

Finally, there has not yet been an extensive evaluation of pure consequence-based procedures that are known to be complete beyond the $\mathcal{EL}$ family. Hence, the practicality of the consequence-based approach for more expressive logics has yet to be determined.

1.3 Contributions

The main objective of this thesis is to develop the first consequence-based calculus for the description logic $\mathcal{ALCHIQ}^+$ (which supports both disjunction and cardinality constraints). The calculus must be such that it is amenable to practical implementation. By using the encoding of role chains by Kazakov [Kaz08], our calculus will also handle $\mathcal{SRIQ}$, which covers all of OWL 2 DL except for nominals and datatypes. A further aim of this thesis is to implement our calculus in a new reasoning system and to evaluate our system against the state-of-the-art.

Consequence-Based Calculus for $\mathcal{SRIQ}$ We present a consequence-based calculus for the DL $\mathcal{ALCHIQ}^+$. Borrowing ideas from resolution theorem proving, we encode the derived con-

sequences as first-order clauses of a specific form, and we handle equality using a variant of *ordered paramodulation* [NR95]—a state-of-the-art calculus for equational theorem proving used in modern theorem provers such as E [Sch02] and Vampire [RV02]. Furthermore, we have carefully constrained the inference rules so that our calculus mimics existing calculi on $\mathcal{ELH}$ ontologies, which ensures robust performance of our calculus on 'mostly-$\mathcal{ELH}$' ontologies. Moreover, our calculus has all the favourable properties of existing consequence-based calculi outlined above, including one-pass classification, worst-case optimality, and pay-as-you-go behaviour.

The Sequoia System and Evaluation We have implemented a system and compared its performance with that of well-established reasoners. We have designed and documented several implementation techniques and data structures that have proven to be crucial to achieving good performance. Our results suggest that our system can comfortably outperform FaCT++, Pellet, and HermiT. Furthermore, due to the relationship between our calculus and existing well-proven calculi, our evaluation shows that our system does indeed perform well on 'mostly-$\mathcal{ELH}$' ontologies. This is the first extensive evaluation of a pure consequence-based procedure that is known to be complete for logics beyond the $\mathcal{EL}$ family.

1.4 Structure and Outline

This thesis details the design and implementation of an algorithm for consequence-based reasoning for $\mathcal{SRIQ}$ ontologies. The remainder of this thesis is organised as follows.

In Part I, we recapitulate the logical foundations that are exploited in this thesis, and review existing work on ontology classification.

- In Chapter 2, we present the preliminaries for the rest of this thesis. First, we recapitulate the definitions and notations of many-sorted clausal logic where equality is the only predicate, and then review the well-known encoding of non-equational atoms into equations. Next, we define the syntax and semantics of the description logic $\mathcal{SRIQ}$ and various sublogics, as well as the reasoning tasks of ontology satisfiability testing and classification. We detail how a $\mathcal{SRIQ}$ ontology in DL-style syntax can be transformed in an equisatisfiable $\mathcal{ALCHIQ}^+$ ontology in DL-style syntax. Finally,-we describe how an $\mathcal{ALCHIQ}^+$ ontology in DL-style syntax is structurally transformed into an equisatisfiable

set of *DL-clauses*. This set is provided as input to our calculus described in Part II.

- In Chapter 3, we survey related work and existing systems. In particular, we discuss tableau, resolution, and consequence-based procedures for related logics. Furthermore, we discuss combined approaches to ontology classification.

In Part II, we present our consequence-based calculus for $\mathcal{ALCHIQ}^+$.

- In Chapter 4, we provide the technical motivation for the design of our calculus, including the anticipated advantages of extending consequence-based reasoning to $\mathcal{ALCHIQ}^+$ when reasoning with ontologies that are challenging for existing tableau- and resolution-based calculi.
- In Chapter 5, we formally present our calculus for $\mathcal{ALCHIQ}^+$. We describe the parameters that can be used to tune reasoning for a particular ontology, including context literal orders and the expansion strategy. Furthermore, we discuss the pay-as-you-go behaviour of our calculus for $\mathcal{ELH}$ and Horn ontologies, and state the worst-case optimal complexity for our calculus. We demonstrate the working of our calculus on examples, and describe some difficulties in handling the interactions between cardinality constraints, inverse roles, and disjunction.
- In Chapter 6, we present all proofs for the results of the previous chapter. In particular, we prove that our calculus is both sound and complete for classification. Our proof of completeness is quite involved, and is based upon rewrite systems from first-order equational theorem proving, and therefore we first provide an outline of the structure of our completeness argument. Furthermore, we show that our calculus is worst-case optimal for $\mathcal{ALCHIQ}^+$. Finally, we show that our calculus is worst-case optimal for $\mathcal{ELH}$ and mimics existing calculi on $\mathcal{ELH}$ ontologies (and thus offers pay-as-you-go behaviour on 'mostly-$\mathcal{ELH}$' ontologies).

In Part III, we describe the Sequoia reasoning system implementing the algorithm detailed in the previous part and provide an extensive evaluation.

- In Chapter 7, we describe the design and implementation of Sequoia, including important details about the saturation algorithm, optimisations to the calculus, redundancy elimination techniques, and data structures.

- In Chapter 8, we present a comprehensive evaluation of Sequoia. First, Sequoia is compared with other state-of-the-art reasoning systems for performance and robustness. Secondly, we conduct experiments to measure the effectiveness of the redundancy elimination techniques presented in Chapter 7.

Finally, in Chapter 9, we conclude with a discussion on our current approach and present possibilities for future work.

Part I

Foundations

Chapter 2

Preliminaries

In this chapter, we recapitulate the standard definitions and results used in the remainder of this thesis. In Section 2.1, we recall many-sorted clausal equational logic. In Section 2.2, we describe how a formula containing predicates can be encoded into an equisatisfiable formula of many-sorted clausal equational logic. In Section 2.3, we define the basic notions of orders. In Section 2.4, we recapitulate the definitions of ground rewrite systems. In Section 2.5, we introduce the syntax and semantics of $\mathcal{SRIQ}$ and the various sublogics that we consider in this thesis. Finally, in Section 2.6, we describe the form of DL-clauses and define how the axioms of a DL ontology are normalised into DL-clauses.

2.1 Many-Sorted Clausal Logic

In this section, we recapitulate the notions from many-sorted clausal equational logic—that is, a clausal logic where equality is the only predicate. This is without loss of generality since, using the well-known encoding shown in Section 2.2, one can always encode non-equational atoms using only the equality predicate [NR01].

First-order clauses are constructed using the symbols of a *many-sorted signature*, which is a pair $\Sigma = (\Sigma^O, \Sigma^F)$ where Σ^O is a non-empty set of *sorts*, and Σ^F is a countable set of function symbols. Each function symbol f is associated with a *symbol type*, which is an expression of the form $o_1 \times \cdots \times o_n \to o$ where $n \geq 0$ and $\{o_1, \ldots, o_n, o\} \subseteq \Sigma^O$ are sorts. Each such f is *of arity* n and *of sort* o, and f is a *constant* if n is zero. For o a sort, set Σ^F_o contains precisely all symbols in Σ^F of sort o. Moreover, we assume that, for each sort o, there is a distinct, countable set of *variables*

of sort o. The set of *terms*, each associated with a sort in Σ^O, is the smallest set of expressions satisfying the following properties:

(i) each variable x of sort o is a term of sort o; and

(ii) if $f \in \Sigma^F$ is a function symbol with symbol type $o_1 \times \cdots \times o_n \to o$ and $t_1, \ldots, t_n$ are terms such that each t_i is of sort o_i, then $f(t_1, \ldots, t_n)$ is a term of sort o.

A term of sort o is also called an *o-term*. An *equality* is an expression of the form $t_1 \approx t_2$ where t_1 and t_2 are terms of the same sort. An *inequality* is the negation of an equality, and $\neg(t_1 \approx t_2)$ is typically written as $t_1 \not\approx t_2$. We assume that $\approx$ and $\not\approx$ are implicitly symmetric—that is, $t_1 \bowtie t_2$ and $t_2 \bowtie t_1$ are one and the same expression, for $\bowtie \in \{\approx, \not\approx\}$. A *literal* is an equality or an inequality. A *clause* is a formula of the form $\forall \vec{x}.[\Gamma \to \Delta]$ where Γ is a conjunction of equalities called the *body*, Δ is a disjunction of literals called the *head*, and $\vec{x}$ contains all variables occurring in the clause; quantifier $\forall \vec{x}$ is usually left implicit. It is common in the literature to assume that clause heads contain equalities only; however, in this thesis it will be convenient to consider clauses whose heads can contain inequalities as well as equalities. We often treat conjunctions and disjunctions as sets (i.e., they are unordered and without repetition) and use them in standard set operations; we write the empty conjunction (disjunction) as $\top$ ($\bot$). A term, literal, clause, or a set thereof is *ground* if it does not contain a variable.

A *substitution* σ is a mapping of variables to terms that is not the identity on a finite number of elements and that respects the sorts—that is, for each variable x, the sorts of x and $\sigma(x)$ are the same. We often write substitutions as $\sigma = \{x_1 \mapsto t_1,\ x_2 \mapsto t_2,\ \ldots\}$ by listing all non-identity mappings. The result of applying a substitution σ to a term or a formula α is defined as usual and is written $\alpha\sigma$. A substitution is *ground* if it maps all variables of its domain to ground terms.

A *position* is a finite sequence of positive integers, written as $i_1.i_2 \ldots i_n$, and where ϵ denotes the empty sequence. Let p be a position and let t be a term. The *subterm of t at position p* is defined inductively as follows: $t|_\epsilon = t$ and, if $t = f(t_1, \ldots, t_n)$, then $t|_{i.p} = t_i|_p$ if $1 \leq i \leq n$ (and is undefined if $i > n$). A position p is *proper* in a term t if $p \neq \epsilon$; and $s[t]_p$ is the term obtained by replacing the subterm of s at position p with t (and this notion is undefined if $s|_p$ an t are of different sorts).

Clauses are commonly interpreted in Herbrand interpretations. The *Herbrand universe* for Σ is the set HU of all ground terms constructed using the symbols of Σ; moreover, for o a sort, HU_o

contains precisely all ground terms of HU of sort o. An *Herbrand equality interpretation* is a set of ground equalities satisfying the usual congruence properties. For α a (not necessarily ground) conjunction, disjunction, clause, or a set thereof, satisfaction of α in an Herbrand equality interpretation I, written $I \models \alpha$, is defined as usual, but with the difference that each variable of sort o quantifies only over the terms in HU_o (rather than all the terms in HU). Note that a ground disjunction of literals Δ may contain inequalities and therefore $I \models \Delta$ does not necessarily imply $I \cap \Delta \neq \emptyset$. Let $\mathcal{O}$ be a set of clauses. Herbrand equality interpretation I is a *model* of $\mathcal{O}$ if $I \models \mathcal{O}$; moreover, $\mathcal{O}$ is *satisfiable* if it admits a model; and finally, $\mathcal{O}$ *entails* a clause $\Gamma \to \Delta$ if $I \models \Gamma \to \Delta$ for each model I of $\mathcal{O}$.

To understand how the presence of sorts affects the semantics of first-order logic, let $\mathcal{O}$ be the set containing clauses $\to x_1 \approx x_2$ and $\to c_1 \not\approx c_2$. Now $\mathcal{O}$ is unsatisfiable if variables x_1 and x_2 are of the same sort as constants c_1 and c_2; otherwise, $\mathcal{O}$ is satisfiable because variables x_1 and x_2 then do not range over the constants c_1 and c_2.

2.2 Encoding Predicates by Function Symbols

Herbrand equality interpretations can be conveniently represented using rewrite systems (see Section 2.4). In such a case, it is convenient to assume that equality is the only predicate, which is without loss of generality since, as we discuss next, each formula containing predicates other than equality can be transformed into an equisatisfiable formula of many-sorted first-order logic where equality is the only predicate [NR01].

To this end, in the rest of this thesis we assume that the signature Σ contains a distinct *predicate* sort $\mathsf{p} \in \Sigma^O$ whose use is restricted such that, for each type $o_1 \times \cdots \times o_n \to o$ of a symbol in the signature Σ, we have $o_i \neq \mathsf{p}$ for each $1 \leq i \leq n$. Finally, we assume that true is a distinct constant of sort p. Then, an atom $P(t_1, \ldots, t_n)$ where P is a predicate other than equality is transformed into the equality $P(t_1, \ldots, t_n) \approx \mathsf{true}$ where P is a function symbol of sort p. For example, given a function symbol P of sort p and a function symbol f of a different sort, both $f(P(x))$ and $P(P(x))$ are not well-formed, whereas $P(f(x))$ is a well-formed p-term.

It is well known that this transformation preserves satisfiability and entailment [NR01]. We write $P(t_1, \ldots, t_n)$ to abbreviate $P(t_1, \ldots, t_n) \approx \mathsf{true}$ whenever the intended meaning is clear from the context, and we call each equality of that form an *atom*.

2.3 Orders

A *strict order* $\succ$ on a set Ω is an irreflexive, asymmetric, and transitive relation on Ω; the corresponding *non-strict order* $\succeq$ is given by defining $a \succeq b$ if and only if $a \succ b$ or $a = b$. An order $\succ$ is *total* if, for all $a, b \in \Omega$, we have $a \succ b$, $b \succ a$, or $a = b$. Given $\circ \in \{\succ, \succeq\}$, element $b \in \Omega$, and subset $S \subseteq \Omega$, the notation $S \circ b$ abbreviates $\exists a \in S : a \circ b$. Given a strict order $\succ$ on a set Ω, we define the *multiset extension* $\succ_{mul}$ of $\succ$ to be the order on the set of all finite multisets over Ω as follows: $M \succ_{mul} N$ if and only if $M \neq N$ and, for each $n \in N \setminus M$, some $m \in M \setminus N$ exists such that $m \succ n$, where $\setminus$ is the multiset difference operator.

A *term order* $\succ$ is a *strict* order on the Herbrand universe HU. We extend $\succ$ to literals by identifying each $s \not\approx t$ with the multiset $\{s, s, t, t\}$ and each $s \approx t$ with the multiset $\{s, t\}$, and by comparing the result using the multiset extension of $\succ$. We reuse the symbol $\succ$ for the induced literal order since the intended meaning should be clear from the context.

2.4 Rewrite Systems

In the proof of the completeness theorem for our calculus presented in Chapter 6, we construct a model of an ontology, which, as is common in equational theorem proving, we represent using a ground *rewrite system*. Hence, we recapitulate the definitions of rewrite systems, following the presentation and the terminology introduced by Baader and Nipkow [BN98]. For simplicity, we adapt all standard definitions to ground rewrite systems only.

A (ground) *rewrite system* R is a binary relation on the Herbrand universe HU. Each pair $(s, t) \in R$ is called a *rewrite rule* and is commonly written as $s \Rightarrow t$. The *rewrite relation* $\rightarrow_R$ for R is the smallest binary relation on HU such that, for all terms $s_1, s_2, t \in \text{HU}$ and each (not necessarily proper) position p in t, if $s_1 \Rightarrow s_2 \in R$, then $t[s_1]_p \rightarrow_R t[s_2]_p$. Moreover, $\xrightarrow{*}_R$ is the reflexive–transitive closure of $\rightarrow_R$, and $\xleftrightarrow{*}_R$ is the reflexive–symmetric–transitive closure of $\rightarrow_R$. A term s is *irreducible by* R if no term t exists such that $s \rightarrow_R t$; and a literal, clause, or substitution α is *irreducible by* R if each term occurring in α is irreducible by R. Moreover, term t is a *normal form* of s with respect to R if $s \xleftrightarrow{*}_R t$ and t is irreducible by R. We consider the following properties of rewrite systems.

- R is *terminating* if no infinite sequence $s_1, s_2, \ldots$ of terms exists such that, for each i, we have

$s_i \to_R s_{i+1}$.

- R is *left-reduced* if, for each $s \Rightarrow t \in R$, the term s is irreducible by $R \setminus \{s \Rightarrow t\}$.
- R is *Church-Rosser* if, for all terms t_1 and t_2 such that $t_1 \overset{*}{\leftrightarrow}_R t_2$, a term z exists such that $t_1 \overset{*}{\to}_R z$ and $t_2 \overset{*}{\to}_R z$.

If rewrite system R is both terminating and left-reduced, then R is Church-Rosser [BN98, Theorem 2.1.5 and Exercise 6.7]. If R is Church-Rosser, then each term s has a unique normal form t such that $s \overset{*}{\to}_R t$ holds. The *Herbrand equality interpretation induced by* a rewrite system R is the set R^* such that, for all $s, t \in \mathrm{HU}$, we have $s \approx t \in R^*$ if and only if $s \overset{*}{\leftrightarrow}_R t$.

Term orders can be used to prove termination of rewrite systems. A term order $\succ$ on ground terms (i.e., on HU) is a *simplification order* if the following conditions hold:

(i) for all ground terms s_1, s_2, and t, and each position p in t, we have that $s_1 \succ s_2$ implies $t[s_1]_p \succ t[s_2]_p$; and

(ii) for each term s and each proper position p in s, we have $s \succ s|_p$.

Given a rewrite system R, if a simplification order $\succ$ exists such that $s \Rightarrow t \in R$ implies $s \succ t$, then R is terminating [BN98, Theorems 5.2.3 and 5.4.8], and, for all ground terms s and t, we have that $s \to_R t$ implies $s \succ t$.

2.5 Description Logics and Inference Problems

In this section, we introduce the syntax and semantics of $\mathcal{SRIQ}$, as well as its logical fragments $\mathcal{ALCHIQ}^+$ and $\mathcal{ELH}$. Furthermore, we state the relevant inference problems and describe the commonly used restrictions on the combinations of the different axioms of $\mathcal{SRIQ}$, which are required to ensure decidability for the inference problems that we are considering.

We base our presentation of $\mathcal{SRIQ}$ on that of Horrocks et al. [HKS06]; for a gentler introduction to description logics, we refer the reader to the Description Logic Primer [KSH12]. In order to simplify the presentation, we assume that ontologies do not contain individuals, since our calculus neither supports nominals nor ABox assertions.

A *signature* is a pair $\Sigma = (\mathsf{N}_C, \mathsf{N}_R)$ consisting of countably infinite and disjoint sets of *atomic concepts* N_C and *atomic roles* N_R. We also distinguish a subset $\mathsf{N}_S \subseteq \mathsf{N}_R$ of *simple atomic roles*.

Table 2.1: Syntax and semantics of $\mathcal{SRIQ}$ concepts and roles

	Syntax	Semantics
Roles:		
atomic role	T	$T^{\mathcal{I}}$
inverse role	T^{-}	$\{ \langle x, y \rangle \mid \langle y, x \rangle \in T^{\mathcal{I}} \}$
universal role	U	$\mathcal{U}^{\mathcal{I}} \times \mathcal{U}^{\mathcal{I}}$
Concepts:		
atomic concept	A	$A^{\mathcal{I}}$
top concept	$\top$	$\mathcal{U}^{\mathcal{I}}$
bottom concept	$\bot$	$\emptyset$
conjunction	$C \sqcap D$	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$
disjunction	$C \sqcup D$	$C^{\mathcal{I}} \cup D^{\mathcal{I}}$
complement	$\neg C$	$\mathcal{U}^{\mathcal{I}} \setminus C^{\mathcal{I}}$
existential restriction	$\exists R.C$	$\{ x \mid \exists y : \langle x, y \rangle \in R^{\mathcal{I}} \wedge y \in C^{\mathcal{I}} \}$
universal restriction	$\forall R.C$	$\{ x \mid \forall y : \langle x, y \rangle \in R^{\mathcal{I}} \rightarrow y \in C^{\mathcal{I}} \}$
at-least restriction	$\geqslant n\, S.C$	$\{ x \mid \sharp\{ y \mid \langle x, y \rangle \in S^{\mathcal{I}} \wedge y \in C^{\mathcal{I}} \} \geq n \}$
at-most restriction	$\leqslant n\, S.C$	$\{ x \mid \sharp\{ y \mid \langle x, y \rangle \in S^{\mathcal{I}} \wedge y \in C^{\mathcal{I}} \} \leq n \}$
local reflexivity	$\exists S.\mathsf{Self}$	$\{ x \mid \langle x, x \rangle \in S^{\mathcal{I}} \}$

Table 2.2: Syntax and semantics of $\mathcal{SRIQ}$ axioms

	Syntax	Semantics
TBox:		
concept inclusion	$C \sqsubseteq D$	$C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$
concept equivalence	$C \equiv D$	$C^{\mathcal{I}} = D^{\mathcal{I}}$
RBox:		
simple role inclusion	$S_1 \sqsubseteq S_2$	$S_1^{\mathcal{I}} \subseteq S_2^{\mathcal{I}}$
role equivalence	$R_1 \equiv R_2$	$R_1^{\mathcal{I}} = R_2^{\mathcal{I}}$
complex role inclusion	$R_1 \circ \cdots \circ R_n \sqsubseteq T$	$R_1^{\mathcal{I}} \circ \cdots \circ R_n^{\mathcal{I}} \subseteq T^{\mathcal{I}}$
role disjointness	$\mathsf{Disjoint}(S_1, S_2)$	$S_1^{\mathcal{I}} \cap S_2^{\mathcal{I}} = \emptyset$
reflexivity	$\mathsf{Reflexive}(R_1)$	$\{ \langle x, x \rangle \mid x \in \mathcal{U}^{\mathcal{I}} \} \subseteq R_1^{\mathcal{I}}$

The set of *roles* over Σ is $\mathbf{R} = \mathsf{N}_R \cup \{ R^- \mid R \in \mathsf{N}_R \}$; roles of the form R^- are called *inverse roles*. The function inv is defined on $\mathbf{R}$ such that $\mathsf{inv}(R) = R^-$ and $\mathsf{inv}(R^-) = R$ where R is an atomic role. The set of *simple roles* is $\mathbf{S} = \{ R \in \mathbf{R} \mid \{R, \mathsf{inv}(R)\} \cap \mathsf{N}_S \neq \emptyset \}$. The set of *concepts* $\mathbf{C}$ over Σ is defined to be the smallest set such that:

(i) $\top$ and $\bot$ are both concepts;

(ii) each atomic concept $A \in \mathsf{N}_C$ is a concept;

(iii) for each role $R \in \mathbf{R}$, if C and D are concepts, then $\neg C$, $C \sqcap D$, $C \sqcup D$, $\exists R.C$, and $\forall R.C$ are concepts; and

(iv) for each simple role $S \in \mathbf{S}$, and each non-negative integer n, if C is a concept, then $\geqslant n\, S.C$, $\leqslant n\, S.C$, and $\exists S.\mathsf{Self}$ are concepts.

The syntax of $\mathcal{SRIQ}$ concepts and roles is summarised in Table 2.1, where $A \in \mathsf{N}_C$ is an atomic concept, $C, D \in \mathbf{C}$ are concepts, $T \in \mathsf{N}_R$ is an atomic role, $R \in \mathbf{R}$ is a role, and $S \in \mathbf{S}$ is a simple role.

An *interpretation* over Σ is a pair $\mathcal{I} = (\mathcal{U}^\mathcal{I}, \cdot^\mathcal{I})$, where $\mathcal{U}^\mathcal{I}$ is a non-empty set called the *domain* of $\mathcal{I}$, and $\cdot^\mathcal{I}$ is an *interpretation function* which associates with each atomic concept A a subset $A^\mathcal{I} \subseteq \mathcal{U}^\mathcal{I}$, and associates with each atomic role R a binary relation $R^\mathcal{I} \subseteq \mathcal{U}^\mathcal{I} \times \mathcal{U}^\mathcal{I}$. The interpretation function $\cdot^\mathcal{I}$ is extended to concepts and roles as shown in Table 2.1.

A *general concept inclusion* (GCI) is of the form $C \sqsubseteq D$ where C and D are concepts. A TBox $\mathcal{T}$ is a finite set of GCIs. We use $C \equiv D$ as an abbreviation for the pair of GCIs $C \sqsubseteq D$ and $D \sqsubseteq C$. An interpretation $\mathcal{I}$ *satisfies* a GCI $C \sqsubseteq D$, written $\mathcal{I} \models C \sqsubseteq D$, if $C^\mathcal{I} \subseteq D^\mathcal{I}$, and it satisfies a TBox $\mathcal{T}$ if, for each $C \sqsubseteq D \in \mathcal{T}$, we have $\mathcal{I} \models C \sqsubseteq D$.

An RBox $\mathcal{R}$ is a finite set of assertions of the following forms:

(i) $S_1 \sqsubseteq S_2$, called a *simple role inclusion*, where S_1 and S_2 are simple roles;

(ii) $R_1 \circ \cdots \circ R_n \sqsubseteq T$, called a *complex role inclusion*, where n is a positive integer, each R_i is a role for $1 \leq i \leq n$, and T is a role which is not simple;

(iii) $\mathsf{Disjoint}(S_1, S_2)$, called *role disjointness*, where S_1 and S_2 are simple roles; or

(iv) $\mathsf{Reflexive}(R)$, called *reflexivity*, where R is any role.

An interpretation $\mathcal{I}$ satisfies a simple role inclusion $S_1 \sqsubseteq S_2$ if $S_1^\mathcal{I} \subseteq S_2^\mathcal{I}$, a complex role inclusion $R_1 \circ \cdots \circ R_n \sqsubseteq U$ if $R_1^\mathcal{I} \circ \cdots \circ R_n^\mathcal{I} \subseteq U^\mathcal{I}$ where $\circ$ denotes the composition of binary relations, a

role disjointness axiom $\mathsf{Disjoint}(S_1, S_2)$ if $S_1^{\mathcal{I}} \cap S_2^{\mathcal{I}} = \emptyset$, and a reflexivity axiom $\mathsf{Reflexive}(R_1)$ if $\{ \langle x, x \rangle \mid x \in \mathcal{U}^{\mathcal{I}} \} \subseteq R_1^{\mathcal{I}}$. An interpretation satisfies an RBox $\mathcal{R}$ if it satisfies all axioms in $\mathcal{R}$. The syntax and semantics of $\mathcal{SRIQ}$ axioms are summarised in Table 2.2, where $C, D \in \mathbf{C}$ are concepts, each $R_i \in \mathbf{R}$ is a role, and $S_1, S_2 \in \mathbf{S}$ are simple roles, and T is a role which is not simple.

The aforementioned constructors allow us to express several assertions on roles.

Symmetry The (either complex or simple) role inclusion $R^- \sqsubseteq R$ asserts that the interpretation of R must be symmetric—that is, $\langle x, y \rangle \in R^{\mathcal{I}}$ implies $\langle y, x \rangle \in R^{\mathcal{I}}$.

Asymmetry The assertion $\mathsf{Disjoint}(S, S^-)$ requires that the interpretation of the simple role S is asymmetric—that is, $\langle x, y \rangle \in S^{\mathcal{I}}$ implies $\langle y, x \rangle \notin S^{\mathcal{I}}$.

Transitivity The complex role inclusion $T \circ T \sqsubseteq T$ asserts that the interpretation of T must be transitive—that is, $\langle x, y \rangle \in T^{\mathcal{I}}$ and $\langle y, z \rangle \in T^{\mathcal{I}}$ together imply $\langle x, z \rangle \in T^{\mathcal{I}}$.

Irreflexivity The concept inclusion $\exists S.\mathsf{Self} \sqsubseteq \bot$ asserts that the interpretation of the simple role S is irreflexive—that is, $\langle x, x \rangle \notin S^{\mathcal{I}}$ for all $x \in \mathcal{U}^{\mathcal{I}}$.

A strict partial order $\prec$ on the set of roles $\mathbf{R}$ is a *regular order* if and only if $R_1 \prec R_2$ whenever $\mathsf{inv}(R_1) \prec R_2$ for all R_1 and R_2. An RBox $\mathcal{R}$ is *regular* if there exists a regular order $\prec$ such that for every simple role inclusion axiom $S_1 \sqsubseteq S_2$ in $\mathcal{R}$ either $S_1 = \mathsf{inv}(S_2)$ or $S_1 \prec S_2$, and for every complex role inclusion axiom $\omega \sqsubseteq T$ in $\mathcal{R}$, one of the following conditions hold:

(i) $\omega = T \circ T$; or

(ii) $\omega = \mathsf{inv}(T)$; or

(iii) $\omega = R_1 \circ \cdots \circ R_n$ and $R_i \prec T$ for all $1 \leq i \leq n$; or

(iv) $\omega = T \circ R_1 \circ \cdots \circ R_n$ and $R_i \prec T$ for all $1 \leq i \leq n$; or

(v) $\omega = R_1 \circ \cdots \circ R_n \circ T$ and $R_i \prec T$ for all $1 \leq i \leq n$.

A $\mathcal{SRIQ}$ ontology in DL-style syntax $\mathcal{K} = \langle \mathcal{T}, \mathcal{R} \rangle$ consists of a TBox $\mathcal{T}$ and a regular RBox $\mathcal{R}$. An interpretation $\mathcal{I}$ is a *model* of $\mathcal{K}$, written $\mathcal{I} \models \mathcal{K}$, if it satisfies the TBox and the RBox of $\mathcal{K}$. In this thesis, we consider the following standard inference problems [BCMN+10], which we now describe. Let $\mathcal{K}$ be a $\mathcal{SRIQ}$ ontology in DL-style syntax.

Concept satisfiability A concept C is *satisfiable* with respect to $\mathcal{K}$ if there exists a model $\mathcal{I}$ of $\mathcal{K}$ such that $C^{\mathcal{I}} \neq \emptyset$.

Concept subsumption A concept C is subsumed by a concept D with respect to $\mathcal{K}$ if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ for every model $\mathcal{I}$ of $\mathcal{K}$. In this case we write $\mathcal{K} \models C \sqsubseteq D$.

Concept equivalence Two concepts C and D are *equivalent* with respect to $\mathcal{K}$ if $C^{\mathcal{I}} = D^{\mathcal{I}}$ for every model $\mathcal{I}$ of $\mathcal{K}$. In this case we write $\mathcal{K} \models C \equiv D$.

Concept disjointness Two concepts C and D are *disjoint* with respect to $\mathcal{K}$ if $C^{\mathcal{I}} \cap D^{\mathcal{I}} = \emptyset$ for every model $\mathcal{I}$ of $\mathcal{K}$.

Classification The *classification* of $\mathcal{K}$ is the set of all entailed subsumptions and equivalences between $\top$, $\bot$, and atomic concepts occurring in $\mathcal{K}$.

Note that, in order to guarantee decidability of the above inference problems for $\mathcal{SRIQ}$, cardinality constraints and local reflexivity are allowed only on simple roles [HST00a]; for similar reasons, role disjointness, irreflexivity, and asymmetry can be asserted only for simple roles. Furthermore, our definition of a $\mathcal{SRIQ}$ ontology in DL-style syntax only permits an RBox that satisfies the above regularity requirements. This condition is also required for decidability of the above inference problems [HS04]. The above inference problems are 2EXPTIME-hard for $\mathcal{SRIQ}$ [Kaz08].

An $\mathcal{ALCHIQ}^+$ ontology in DL-style syntax is a $\mathcal{SRIQ}$ ontology in DL-style syntax which contains no complex role inclusion axioms. In the absence of complex role inclusion axioms, there is no distinction between simple and non-simple roles. Hence, in the context of $\mathcal{ALCHIQ}^+$ we assume that all roles are simple. The above inference problems are EXPTIME-complete for $\mathcal{ALCHIQ}^+$ [LST05, Tob01].

An $\mathcal{ELH}$ ontology in DL-style syntax is a $\mathcal{ALCHIQ}^+$ ontology in DL-style syntax in which concepts are only constructed from atomic concepts, the top concept, conjunction and existential restriction, and where the RBox only contains simple role inclusions. Furthermore, an $\mathcal{EL}$ ontology in DL-style syntax is an $\mathcal{ELH}$ ontology in DL-style syntax in which the RBox is empty. The above inference problems are PTIME-complete for $\mathcal{EL}$ and $\mathcal{ELH}$ [BBL05].

A $\mathcal{SRIQ}$ ontology in DL-style syntax can be transformed into an equisatisfiable $\mathcal{ALCHIQ}^+$ ontology in DL-style syntax by eliminating complex role inclusion axioms and simulating their

effects using additional concept inclusion axioms. The encoding makes use of the standard translation of complex role inclusion axioms into finite automata [HKS06, DdN05]. This reduction allows us to simplify the algorithm we present in this thesis by only considering $\mathcal{ALCHIQ}^+$ ontologies without loss of generality. Unfortunately, the size of the ontology may increase exponentially as a result of this transformation, where size is measured by the number of symbols required to encode the ontology on the input tape of a Turing machine [HS04]. Such an exponential blowup when converting from $\mathcal{SRIQ}$ to $\mathcal{ALCHIQ}^+$ is unavoidable in the worst-case due to the difference in the complexity of satisfiability testing for the two logics.

2.6 DL-Clauses

Our calculus is based on resolution, and so as input it takes a DL ontology represented as a set of *DL-clauses* constructed using a signature $\Sigma = (\Sigma^O, \Sigma^F)$ where Σ^O consists of the predicate sort p and the *abstract sort* a, and Σ^F contains *abstract concepts* B_i of type $\mathsf{a} \to \mathsf{p}$, *abstract roles* S_i of type $\mathsf{a} \times \mathsf{a} \to \mathsf{p}$, and *abstract successor functions* f_j of type $\mathsf{a} \to \mathsf{a}$. Finally, DL-clauses are built using the *central variable* x and *neighbour variables* z_j distinct from x, all of sort a. A *DL-*a*-term* has the form x, z_j, or $f_j(x)$. A term of the form $f_j(t)$ is called an f_j*-successor* of t, and t is called the *predecessor* of $f_j(t)$. A *DL-*p*-term* has the form $B_i(z_j)$, $B_i(x)$, $B_i(f_j(x))$, $S_i(z_j, x)$, $S_i(x, z_j)$, $S_i(x, x)$, $S_i(f_j(x), x)$, or $S_i(x, f_j(x))$. A *DL-literal* is of the form $A \approx \mathsf{true}$ where A is a DL-p-term, or of the form $l \bowtie r$ where l and r are DL-a-terms and $\bowtie \in \{\approx, \not\approx\}$. A *DL-clause* contains only atoms of the form $B_i(x)$, $S_i(x, x)$, $S_i(z_j, x)$, and $S_i(x, z_j)$ in the body and only DL-literals in the head, and each variable z_j occurring in the clause also occurs in the body. An *ontology* $\mathcal{O}$ is a finite set of DL-clauses. A *query clause* is a DL-clause in which all literals are of the form $B_i(x)$ or $S_i(x, x)$. Given an ontology $\mathcal{O}$ and a query clause $\Gamma \to \Delta$, the computational problem we consider in this thesis is to decide whether $\mathcal{O} \models \Gamma \to \Delta$ holds.

An $\mathcal{ALCHIQ}^+$ ontology in DL-style syntax can be transformed into a set of DL-clauses without affecting query clause entailment as shown in Table 2.3. First, we *normalise* DL axioms to the form shown on the left-hand side of Table 2.3 (i.e., we replace all complex concepts with fresh atomic ones). The normalisation process is well understood [MSH09]. Each DL axiom is then transformed into a DL-clause as shown on the right-hand side of Table 2.3. As usual, in the translation of $B_1 \sqsubseteq \geqslant n\, S.B_2$ shown in the table, we require each f_i to be fresh for $1 \leq i \leq n$. This trans-

Table 2.3: Translating normalised $\mathcal{ALCHIQ}^+$ ontologies into DL-clauses

DL1	$\prod_{1\leq i\leq n} B_i \sqsubseteq \bigsqcup_{n+1\leq i\leq m} B_i$	$\rightsquigarrow$	$\bigwedge_{1\leq i\leq n} B_i(x) \rightarrow \bigvee_{n+1\leq i\leq m} B_i(x)$	
DL2	$B_1 \sqsubseteq \geqslant n\, S.B_2$	$\rightsquigarrow$	$B_1(x) \rightarrow S(x, f_i(x))$	for $1 \leq i \leq n$
			$B_1(x) \rightarrow B_2(f_i(x))$	for $1 \leq i \leq n$
			$B_1(x) \rightarrow f_i(x) \not\approx f_j(x)$	for $1 \leq i < j \leq n$
DL3	$\exists S.B_1 \sqsubseteq B_2$	$\rightsquigarrow$	$S(z_1, x) \wedge B_1(x) \rightarrow B_2(z_1)$	
DL4	$B_1 \sqsubseteq \leqslant n\, S.B_2$	$\rightsquigarrow$	$S(z_1, x) \wedge B_2(x) \rightarrow S_{B_2}(z_1, x)$	for fresh S_{B_2}
			$B_1(x) \wedge \bigwedge_{1\leq i\leq n+1} S_{B_2}(x, z_i) \rightarrow \bigvee_{1\leq i<j\leq n+1} z_i \approx z_j$	
DL5	$S_1 \sqsubseteq S_2$	$\rightsquigarrow$	$S_1(z_1, x) \rightarrow S_2(z_1, x)$	
DL6	$S_1 \sqsubseteq S_2^-$	$\rightsquigarrow$	$S_1(z_1, x) \rightarrow S_2(x, z_1)$	
DL7	$\mathsf{Disjoint}(S_1, S_2)$	$\rightsquigarrow$	$S_1(z_1, x) \wedge S_2(z_2, x) \rightarrow \bot$	
DL8	$B \sqsubseteq \exists S.\mathsf{Self}$	$\rightsquigarrow$	$B(x) \rightarrow S(x, x)$	
DL9	$\exists S.\mathsf{Self} \sqsubseteq B$	$\rightsquigarrow$	$S(x, x) \rightarrow B(x)$	

formation uses the well-known correspondence between DLs and first-order logic [BCMN+10]; however, the standard translation of $B_1 \sqsubseteq \leqslant n\, S.B_2$ requires atoms of the form $B_2(z_i)$ in clause bodies, which are not allowed in DL-clauses. We address this issue by introducing a fresh role S_{B_2} that we axiomatise as $S(z_1, x) \wedge B_2(x) \rightarrow S_{B_2}(z_1, x)$; this subsequently allows us to clausify the original axiom as if it were $B_1 \sqsubseteq \leqslant n\, S_{B_2}$. Moreover, assuming unary encoding of number restrictions, this can be performed in polynomial time. For an $\mathcal{ELH}$ ontology, $\mathcal{O}$ contains DL-clauses of type DL1 with $m = n + 1$, DL2 with $n = 1$, DL3, and DL5.

Note that DL-clauses are more general than what is required for $\mathcal{SRIQ}$. For example, a DL-clause of the form $R(z_1, x) \wedge A(x) \rightarrow S(z_1, x)$ expresses a form of relativized role inclusion, and a DL-clause of the form $R(x, z_1) \wedge S(z_1, x) \rightarrow U(x, z_1) \vee T(z_1, x)$ captures a safe role expression, neither of which can be expressed in $\mathcal{SRIQ}$ [Tob01].

Chapter 3

Ontology Classification

In this chapter, we discuss existing approaches to ontology classification. In Section 3.1, we discuss the early work in this field. Then, in Sections 3.2 and 3.3, we give an overview of the current tableau- and resolution-based approaches to ontology classification, and discuss some of their limitations. In Section 3.4, we discuss existing consequence-based procedures. Finally, in Section 3.5, we review recent approaches that combine different procedures for ontology classification.

3.1 Historical Perspective

Early attempts at knowledge representation were driven by cognitive intuitions and tried to mimic the representation structures and the processes that may participate in human reasoning. Many early systems were based on graphical diagrams whose arrows represented various inheritance, membership and meronymy relations. Such network-based systems [Leh92] were criticised for their lack of precise semantics and hence ambiguity. This was seen as a serious obstacle and the attention of the KRR community shifted towards logic-based formalisms which enjoy more formal extensional semantics. Gradually, these evolved into the large family of DLs.

Amongst the earliest DLs we find $\mathcal{FL}^-$ supporting conjunctions, universal restrictions and limited existential restrictions as considered by Brachman and Levesque [BL84], who showed that $\mathcal{FL}^-$ admits a polynomial-time concept subsumption algorithm (with respect to the empty TBox) based on a purely structural comparison of concept definitions. Although such structural algorithms tend to be incomplete for more expressive DLs, they might still be used to detect

some subsumptions and thus improve performance of a classification algorithm.

An important step in overcoming the incompleteness of reasoning was the publication by Schmidt-Schauß and Smolka of the first tableau-based algorithm for $\mathcal{ALC}$ [SS91], a DL with existential and universal restrictions and all Boolean constructors. The algorithm exploits the model-based semantics of DLs and checks satisfiability of a concept by systematically trying to construct a canonical model. If no such canonical model can be found, the concept is unsatisfiable. It proved to be relatively easy to adapt the algorithm to accommodate new constructors (although the challenge was usually in ensuring termination) and the method was gradually extended to more expressive logics [HST00a, HKS06].

Many researchers were sceptical about the practical applicability of the tableau method. Since reasoning in $\mathcal{ALC}$ is provably intractable [SS91, Tob01], it was believed that DLs with such expressive power were impractical for large-scale applications. However, following the success of propositional SAT-solvers, the implementation of the FaCT reasoner by Horrocks demonstrated that a highly optimised tableau algorithm can provide acceptable behaviour on realistic ontologies [Hor97, Hor98]. Consequently, the focus of the research community moved towards very expressive DLs, such as $\mathcal{SHIQ}$ [HS04], $\mathcal{SHOIQ}$ [HS07], and $\mathcal{SROIQ}$ [HKS06] that form the logical basis of OWL 2 DL, and a number of optimised implementations of the tableau (or related hypertableau) procedure for these logics emerged: FaCT++ [TH06], Racer [HM01], Pellet [SPCK+07] and HermiT [MSH09] are some of the most widely used reasoners available today.

Since DLs are fragments of first-order logic, it is in theory possible to use the satisfiability testing procedure for the corresponding first-order logic fragment. For example, the DL $\mathcal{ALC}$ is contained within the *guarded fragment of first-order logic* [ANvB98], and although $\mathcal{SHIQ}$ provides a restricted form of counting that does not fall within the guarded fragment, it is instead captured by the *two-variable fragment with counting* ($\mathcal{C}^2$) [GOR97]. However, for $\mathcal{SHIQ}$, this provides us with neither a practical nor a worst-case optimal reasoning procedure since satisfiability testing in $\mathcal{C}^2$ is NEXPTIME-complete, whereas satisfiability testing in $\mathcal{SHIQ}$ is EXPTIME-complete. This has further motivated the design of reasoning algorithms specifically for DLs.

The availability of efficient reasoning services for DLs led to the increase of their popularity and, eventually, to the development of very large ontologies. The life-science ontologies, such as the GALEN medical knowledge base [RR06], SNOMED (Systematised Nomenclature of Medicine) [Spa00], FMA (Foundational Model of Anatomy) [RM03], GO (Gene Ontol-

ogy) [ABBB+00], and OBO (Open Biological and Biomedical Ontologies) [SARB+07], have been of particular interest to the KRR community. Comprising tens or even hundreds of thousands of highly interlinked concepts, some of these ontologies are notoriously hard to classify even for state-of-the-art tableau-based reasoners and have been a major driving force behind the more recent developments in ontology classification research.

An example of such development was the increasing focus on $\mathcal{EL}$, a DL which allows only for conjunction and existential restrictions [Baa03]. It was shown that $\mathcal{EL}$ ontologies can be classified in polynomial time by a consequence-based algorithm which computes the closure of the subsumption relations under a certain set of inference rules. Furthermore, the relevance of $\mathcal{EL}$ was established by the discovery that SNOMED, and large parts of GO and GALEN, can be expressed in $\mathcal{EL}$, and that even a relatively unoptimised implementation of the consequence-based algorithm outperforms highly optimised tableau reasoners on these ontologies [BLS05]. Subsequent research focused on identifying maximal tractable extensions of $\mathcal{EL}$ such as $\mathcal{EL}^{++}$, which also supports nominals, role inclusions, and a restricted form of concrete domains [BBL05]. OWL 2 EL is a fragment of OWL 2 based on $\mathcal{EL}^{++}$ [MCHW+09].

$\mathcal{EL}^{++}$ sacrifices many common constructors in order to guarantee tractability of reasoning. However, the underlying consequence-based method can be extended to superpolynomial logics, and this is the topic of Section 3.4. First, however, we discuss tableau- and resolution-based procedures for more expressive logics in Sections 3.2 and 3.3.

3.2 Tableau

Tableau and hypertableau are currently the most commonly used algorithms for reasoning with ontologies. The basic reasoning task supported by tableau-based reasoners is concept satisfiability testing. In order to show that a concept is satisfiable, a tableau-based reasoner systematically attempts to construct (the representation of) a model of the concept with respect to the ontology [vHLP07, Chapter 3].

A partially constructed interpretation is represented by a set of assertions of the form $C(u)$ and $R(u,v)$, where C is a (possibly complex) concept expression, R is a role, and u and v are elements of the domain of the interpretation under construction. The intuitive meaning of the concept assertion $C(u)$ is that u is an instance of C, and the intuitive meaning of the role assertion

$R(u, v)$ is that the element u is R-related to the element v in the model. The partially constructed model is repeatedly extended using a number of so-called *expansion rules*. The expansion rules operate directly on the assertions, and syntactically decompose the concept assertions to infer new assertions.

We now briefly illustrate the behaviour of the tableau calculus presented by Horrocks and Sattler [HS07] for $\mathcal{ALCHIQ}$ ontologies. For example, if $(C_1 \sqcap C_2)(u)$ holds then the $\sqcap$-rule ensures that both $C_1(u)$ and $C_2(u)$ hold. However, the $\sqcup$-rule that handles disjunctions between concepts is *nondeterministic*: if $(C_1 \sqcup C_2)(u)$ and neither $C_1(u)$ nor $C_2(u)$ hold, then the algorithm will first try to construct a model in which either $C_1(u)$ or $C_2(u)$ holds, and if that construction leads to an immediate contradiction in the assertions (such as $D(u)$ and $\neg D(u)$), the algorithm will backtrack and construct a model in which the other assertion holds instead. The construction continues until a model has been constructed or an inconsistency is found.

Moreover, given the assertion $\exists R.C(u)$, the $\exists$-rule ensures that there exists a fresh element v such that $R(u, v)$ and $C(v)$. The $\exists$-rule causes all models that are considered by the tableau algorithm to be tree-shaped. This feature relies on the tree-model property [Var97] (and its relaxations for more expressive DLs [HS07]), which states that every consistent ontology has a tree-shaped model. It allows the algorithms to restrict the search for candidate models to tree-like structures without losing completeness.

For each GCI $C_1 \sqsubseteq C_2$ in the ontology, the $\sqsubseteq$-rule adds a disjunctive assertion $(\neg C_1 \sqcup C_2)(u)$ for each element u. This results in a major source of nondeterminism in the calculus. Numerous optimisations have been developed in an effort to reduce the size of the search space explored by the algorithm as a result of applying nondeterministic tableau expansion rules, including *lazy unfolding*, *semantic branching search*, *local simplification*, and *dependency directed backtracking* [Hor10, THP07]. In particular, various *absorption* optimisations have been developed that attempt to limit the nondeterminism that results from the $\sqsubseteq$-rule [Hor98, TH04, HW06, Hor10]. Alternatively, the hypertableau calculus of Motik et al. [MSH09], which can be seen as a combination of tableau and hyperresolution, generalises all known absorption variants and is guaranteed to exhibit no nondeterminism with Horn DLs, and thus has comparatively far better performance on Horn ontologies.

Termination of the construction is ensured by using a *blocking* mechanism, which prevents expansion rules from being applicable to blocked elements [BS06, BDS93, BBH96]. The block-

ing conditions required depend on the logic that the tableau algorithm supports. For example, tableau algorithms for DLs without inverse roles can use so-called *single subset blocking*, which stops the expansion of a branch of the tree-shaped model whenever the same element label recurs in the branch. However, for DLs that support both inverse roles and cardinality constraints, a different blocking condition known as *pairwise blocking* is necessary [HST00b].

When using a tableau-based algorithm, subsumption testing is reduced to satisfiability testing: $\mathcal{O} \models C \sqsubseteq D$ holds if and only if $C \sqcap \neg D$ is unsatisfiable with respect to $\mathcal{O}$. Classification of an ontology is performed by iterating over all necessary pairs of concepts and performing a subsumption test. Techniques have been developed to try to reduce the number of satisfiability tests required to determine all possible subsumptions [BHNP+94, GHMS+12].

Existing systems using a tableau algorithm include FaCT++ [TH06], Pellet [SPCK+07], and Racer [HM01]. The HermiT system uses hypertableau [MSH09].

3.2.1 Limitations of Tableau-Based Reasoners

We now discuss some of the known limitations of tableau-based calculi. Two main factors that affect the performance of tableau-based reasoners have been identified in the literature [Don10], and are known as *or-branching* (i.e., the degree of nondeterminism) and *and-branching* (i.e., the size of the constructed models).

Or-Branching As previously discussed, tableau calculi handle disjunctive assertions nondeterministically: given $(C \sqcup D)(u)$, if assuming that $C(u)$ holds leads to a contradiction, the algorithm must backtrack and assume that $D(u)$ holds. To show the unsatisfiability of a concept with respect to an ontology (as is required for subsumption testing), every possible guess must lead to a contradiction, which can give rise to exponential behaviour in practice. Backtracking causes the algorithm to discard the parts of the model that were constructed after the last branching point, even though these might be reconstructed again in future branches of the computation. Although the hypertableau algorithm generalises all known absorption variants [MSH09], disjunctions occurring in complex concept expressions in the input ontology can still result in costly or-branching.

And-Branching The expansion of a tree-shaped model due to existential quantifiers can generate very large models, particularly for ontologies whose definitions are highly interconnected and contain cycles [Kaz09].

Several blocking conditions have been proposed to reduce and-branching [MSH09, GHM10]. However, the blocking conditions can be quite strict (for example, pairwise blocking [HS99a]) and an extensive construction is often needed until the conditions are satisfied and all infinite paths of the model become blocked, particularly when there are a large number of cycles in the definitions of concepts in the ontology. Research into performance optimisations include work on improved blocking techniques to reduce the size of the models generated (for example, core blocking [GHM10]) and reusing previously introduced elements (known as *individual reuse* [MH08]).

In addition to memory consumption problems, and-branching can increase or-branching by increasing the number of elements to which GCIs are applied [MSH09].

Other Issues Although worst-case optimal tableau algorithms are known [GN07], the tableau algorithms typically implemented in practical systems not worst-case optimal. This includes the hypertableau calculus implemented in the HermiT reasoner [MSH09]. Furthermore, tableau calculi do not support one-pass classification, which is particularly important for the efficient classification of ontologies where the number of entailed subsumptions is only a small proportion of the total number of possible subsumptions, as is the case for the SNOMED CT ontology [KKS13].

3.3 Resolution

Since DLs are decidable fragments of first-order logic, existing first-order theorem proving techniques can be used for DL reasoning. Resolution has been used as a decision procedure for fragments of modal and first-order logic [dNSH00], as well as for $\mathcal{ALC}$ [HS99b], $\mathcal{SHIQ}$ [HMS08], $\mathcal{SHOIQ}$ [KM08a], and other DLs which fall within a fragment of first-order logic such as the guarded fragment [KdN04, GdN99, dNdR03] or the two-variable fragment [dNP01].

Resolution-based decision procedures work by deriving new clauses that are consequences of the original clauses, in a way that is refutation complete (i.e., the empty clause will be derived if the input set of clauses is unsatisfiable) and such that the saturation terminates.

The basic principles for deciding a fragment $\mathcal{L}$ of first-order logic by a resolution-based procedure were first described in [Joy76]. The general idea is to identify a syntactic form of clauses $\mathcal{N}_\mathcal{L}$ and a calculus $\mathcal{C}$ such that:

(i) every clause C in the fragment $\mathcal{L}$ can be translated into an equisatisfiable set of clauses from $\mathcal{N}_\mathcal{L}$;

(ii) the calculus $\mathcal{C}$ is sound and refutation complete for sets of clauses of the form $\mathcal{N}_\mathcal{L}$;

(iii) for a finite signature, $\mathcal{N}_\mathcal{L}$ contains finitely many clauses up to variable renaming; and

(iv) every inference of $\mathcal{C}$ with premises from $\mathcal{N}_\mathcal{L}$ does not extend the signature and produces only clauses from $\mathcal{N}_\mathcal{L}$.

These conditions guarantee that every saturation produces at most finitely many clauses and hence terminates [Kaz08, FLHT01].

Hence, resolution-based procedures for DLs translate DL axioms into first-order clauses and apply specific resolution strategies which ensure that only a bounded number of clauses are derived, and thus guarantee termination. Resolution calculi allow for many optimisations, such as ordering restrictions and redundant clause deletion [BG01]. KAON2 is a DL reasoner which implements an optimal resolution-based procedure for $\mathcal{SHIQ}$ [HMS08].

Resolution-based procedures for DLs are often (but not always) worst-case optimal [Kaz08], but as we discuss in the next section and in Chapter 4, even in very simple cases they can draw unnecessary inferences.

Unlike consequence-based procedures, but like the tableau approach, a resolution-based reasoner performs subsumption testing by checking inconsistency of a (suitably extended) set of clauses. The classification task is thus reduced to repeated subsumption testing.

3.3.1 Limitations of Resolution

Although resolution-based procedures are usually worst-case optimal, in practice most systems are tableau- or consequence-based. The reason seems to be that, despite optimisations, resolution still performs too many inferences. In the following example, DL axioms are shown on the left

and their translations into first-order logic are shown on the right.

$$A_1 \sqsubseteq \exists R.B_1 \quad \rightsquigarrow \quad A_1(x) \rightarrow R(x, f_1(x)) \tag{3.1}$$

$$A_1(x) \rightarrow B_1(f_1(x)) \tag{3.2}$$

$$\exists R.B_2 \sqsubseteq A_2 \quad \rightsquigarrow \quad R(x,y) \wedge B_2(y) \rightarrow A_2(x) \tag{3.3}$$

Note that these two DL axioms are unrelated except for having a common role. Although they do not interact in tableau- and consequence-based procedures, a resolution-based prover will usually resolve clauses (3.1) and (3.3) to produce the clause $A_1(x) \wedge B_2(f_1(x)) \rightarrow A_2(x)$. In general, such interactions can result in a quadratic number of redundant inferences.

Ontologies often contain a large number of DL axioms of the form shown above, since both axioms are expressible in $\mathcal{EL}$. Moreover, many ontologies contain a large number of axioms involving existential restrictions, but only very few roles [KKS13]. Examples of this include the medical ontologies SNOMED CT and GALEN. Furthermore, an evaluation of KAON2 [HMS04], a reasoner based on basic superposition, demonstrated that the system was not able to classify these and other medical ontologies [Sun09].

Finally, resolution-based procedures do not support one-pass classification and hence, similarly to tableau-based procedures, even if each subsumption test is very efficient, the large number of tests required to construct a subsumption hierarchy can make classification every inefficient overall [GHMS+12].

3.4 Consequence-Based Reasoning

The underlying goal in the development of the $\mathcal{EL}$ family of logics, for which consequence-based procedures were originally developed, was to ensure tractability of reasoning [Baa03, BBL05]. Although adding disjunction to $\mathcal{EL}$ increases the complexity of subsumption checking from PTIME to EXPTIME [BBL05], consequence-based procedures have been successfully developed for $\mathcal{ALCH}$ [SKH11] and $\mathcal{ALCI}$ [SMH14]. Similarly, adding cardinality constraints to $\mathcal{EL}$ raises the complexity of subsumption checking to EXPTIME [BBL05], and consequence-based procedures have also been developed for the logics Horn-$\mathcal{SHIQ}$ [Kaz09] and Horn-$\mathcal{SROIQ}$ [ORS10].

Unlike tableau-based reasoners, consequence-based procedures do not construct models di-

$$\textbf{IR1}\ \frac{}{A \sqsubseteq A} \qquad \textbf{IR2}\ \frac{}{A \sqsubseteq \top} \qquad \textbf{CR1}\ \frac{A \sqsubseteq B}{A \sqsubseteq C} : B \sqsubseteq C \in \mathcal{O}$$

$$\textbf{CR2}\ \frac{A \sqsubseteq B \quad A \sqsubseteq C}{A \sqsubseteq D} : B \sqcap C \sqsubseteq D \in \mathcal{O} \qquad \textbf{CR3}\ \frac{A \sqsubseteq B}{A \sqsubseteq \exists R.C} : B \sqsubseteq \exists R.C \in \mathcal{O}$$

$$\textbf{CR4}\ \frac{A \sqsubseteq \exists R.B \quad B \sqsubseteq C}{A \sqsubseteq D} : \exists R.C \sqsubseteq D \in \mathcal{O} \qquad \textbf{CR10}\ \frac{A \sqsubseteq \exists R.B}{A \sqsubseteq \exists S.B} : R \sqsubseteq S \in \mathcal{O}$$

Figure 3.1: The completion rules for reasoning in $\mathcal{EL}\mathcal{H}$ [BBL05]

rectly but instead derive logical *consequences* of the ontology. For example, Figure 3.1 shows the consequence-based inference rules for $\mathcal{EL}\mathcal{H}$ by Baader et al. [BBL05], which are often referred to as the *completion rules* for $\mathcal{EL}\mathcal{H}$. These inference rules are complete for classification of an $\mathcal{EL}\mathcal{H}$ ontology, that is, the rules derive all subsumptions between atomic concepts that are entailed by the input ontology. The fact that all subsumptions are derived in one saturation, known as *one-pass classification*, makes the consequence-based approach particularly suitable for the classification task.

In Chapter 4, where we present the technical overview of the design of our calculus for $\mathcal{ALCHIQ}^{+}$, we will explain how the derived consequences are not stored altogether in one set, and that the consequences are instead partitioned into multiple *contexts*. Furthermore, we will explain how the inferences between conclusions in different contexts can be restricted, while still satisfying existential and universal restrictions.

Extensive evaluation has demonstrated that consequence-based systems for the $\mathcal{EL}$ family of logics can outperform other highly optimised reasoners [GBJM+13]. Even prototypical implementations of consequence-based procedures for more expressive logics have shown very competitive performance [BLS05, Kaz09, SKH11, KKS11].

Furthermore, systems that implement consequence-based calculi that support expressivity beyond $\mathcal{EL}$ may prove important in real-world applications. For example, a new version of the SNOMED CT anatomical model that uses disjunction is being developed by the International Health Terminology Standards Development Organisation [SKH11].

The main advantage of tableau- and resolution-based reasoners over the existing consequence-based procedures is their universal applicability; no existing consequence-based procedure supports both disjunctions and cardinality constraints.

3.5 Combined Approaches to Reasoning

In addition to the tableau, resolution and consequence-based procedures for reasoning with ontologies, combined approaches to reasoning have also been proposed. In Section 3.5.1, we discuss some existing work on combining reasoning procedures in a black-box fashion, and in Section 3.5.2, we review an approach that tightly couples a saturation-based procedure with tableau.

3.5.1 Black-Box Combination of Reasoning Procedures

Several systems have been developed that combine existing reasoners in a black-box fashion, including MORe [ACH12], Chainsaw [TP12], and WSClassifier [SSD13]. These approaches are independent of the implementation details of the reasoners that they combine, and are independent of the reasoning algorithms that are used.

For example, the reasoning system MORe combines a fully-fledged reasoner together with an efficient reasoner for a given logical fragment in a black-box manner [ACH12]. Module extraction [SPS09] is used to extract the part of an ontology that can be completely handled by a more efficient reasoning system and the fully-fledged reasoner is then only used for the remaining part of the ontology. Since the module for the efficient reasoner has to be reduced as long as the module contains unsupported features, this may affect the pay-as-you-go performance. Finally, module extraction can take a significant amount of time for certain ontologies.

Although it is possible to choose the reasoning systems to be combined based upon what is likely to give better overall reasoning performance for a particular ontology, most existing fully-fledged reasoners are tableau-based, and most existing systems for specific logical fragments are consequence-based.

3.5.2 Tightly Coupled Reasoning Procedures

The Konclude reasoner by Steigmiller and Glimm [SG15] implements an algorithm that tightly integrates a saturation-based procedure with tableau. Specifically, the saturation-based procedure is executed in a first-stage to derive sound consequences of the ontology. In addition, the reasoner keeps track of the possible sources of incompleteness that are then subsequently handled by the tableau. The approach also includes a mechanism whereby information is exchanged between the tableau and the consequence-based procedure, and thus allows the conse-

quence-based procedure to be used again during later stages of reasoning. This technique has proved very effective in practice. However, the algorithm implemented in Konclude does not provide one-pass classification and it is not worst-case optimal for DLs beyond the $\mathcal{EL}$ family.

Part II

Consequence-Based Reasoning

Chapter 4

Overview

In this chapter, we motivate the need for consequence-based calculi and present an overview of our approach. We will do this by example, introducing only as much technical detail as is necessary to understand the examples. We reserve the formal presentation of our algorithm for Chapter 5, and present the proof of its correctness in Chapter 6.

In Section 4.1 we discuss certain inefficiencies of existing DL reasoning calculi, and then in Section 4.2 we discuss how existing consequence-based calculi address these problems by separating clauses into contexts in a way that considerably reduces the number of inferences. Next, in Section 4.3 we discuss the main contribution of this thesis, which lies in extending the consequence-based framework to a DL with disjunctions and cardinality constraints. The latter is handled by equality reasoning, which requires a more involved calculus and completeness proof.

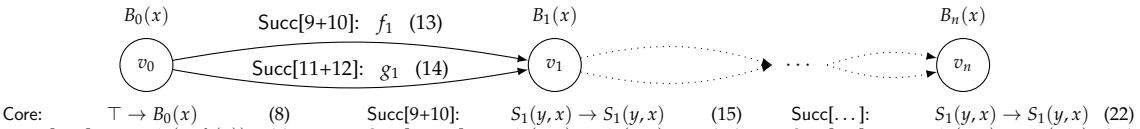
4.1 Why Consequence-Based Calculi?

Consider the $\mathcal{EL}$ ontology $\mathcal{O}_1$ in Figure 4.1; one can readily check that $\mathcal{O}_1 \models B_i(x) \rightarrow C_i(x)$ holds for $0 \leq i \leq n$. We next discuss how both (hyper)tableau and resolution—DL reasoning calculi commonly used in practice—prove these entailments. We show how each of the two calculi can draw unnecessary inferences, and we discuss how consequence-based calculi overcome these inefficiencies by combining the two styles of reasoning.

To prove $\mathcal{O}_1 \models B_0(x) \rightarrow C_0(x)$ using the (hyper)tableau calculus, we start with a fresh constant a and from $B_0(a)$ apply (1) through (7) in a forward-chaining manner. Since $\mathcal{O}_1$ contains

Ontology $\mathcal{O}_1$

$B_i \sqsubseteq \exists S_1.B_{i+1}$	$\rightsquigarrow$	$B_i(x) \rightarrow S_1(x, f_{i+1}(x))$	(1)	for $0 \leq i < n$
		$B_i(x) \rightarrow B_{i+1}(f_{i+1}(x))$	(2)	
$B_i \sqsubseteq \exists S_2.B_{i+1}$	$\rightsquigarrow$	$B_i(x) \rightarrow S_2(x, g_{i+1}(x))$	(3)	
		$B_i(x) \rightarrow B_{i+1}(g_{i+1}(x))$	(4)	
$B_n \sqsubseteq C_n$	$\rightsquigarrow$	$B_n(x) \rightarrow C_n(x)$	(5)	
$\exists S_1.C_{i+1} \sqsubseteq C_i$	$\rightsquigarrow$	$S_1(z_1, x) \wedge C_{i+1}(x) \rightarrow C_i(z_1)$	(6)	for $0 \leq i < n$
$\exists S_2.C_{i+1} \sqsubseteq C_i$	$\rightsquigarrow$	$S_2(z_1, x) \wedge C_{i+1}(x) \rightarrow C_i(z_1)$	(7)	



Core:	$\top \rightarrow B_0(x)$	(8)	Succ[9+10]:	$S_1(y,x) \rightarrow S_1(y,x)$	(15)	Succ[...]:	$S_1(y,x) \rightarrow S_1(y,x)$	(22)
Hyper[1+8]:	$\top \rightarrow S_1(x, f_1(x))$	(9)	Succ[11+12]:	$S_2(y,x) \rightarrow S_2(y,x)$	(16)	Succ[...]:	$S_2(y,x) \rightarrow S_2(y,x)$	(23)
Hyper[2+8]:	$\top \rightarrow B_1(f_1(x))$	(10)	Core:	$\top \rightarrow B_1(x)$	(17)	Core:	$\top \rightarrow B_n(x)$	(24)
Hyper[3+8]:	$\top \rightarrow S_2(x, g_1(x))$	(11)	Hyper[1+17]:	$\top \rightarrow S_1(x, f_2(x))$	(18)	Hyper[5+24]:	$\top \rightarrow C_n(x)$	(25)
Hyper[4+8]:	$\top \rightarrow B_1(g_1(x))$	(12)	Hyper[2+17]:	$\top \rightarrow B_2(x, f_2(x))$	(19)	Hyper[6+22+25]:	$S_1(y,x) \rightarrow C_{n-1}(y)$	(26)
Pred[29]:	$\top \rightarrow C_0(x)$	(31)	Hyper[3+17]:	$\top \rightarrow S_2(x, g_2(x))$	(20)	Hyper[7+23+25]:	$S_2(y,x) \rightarrow C_{n-1}(y)$	(27)
			Hyper[4+17]:	$\top \rightarrow B_2(x, g_2(x))$	(21)			
			Pred[...]:	$\top \rightarrow C_1(x)$	(28)			
			Hyper[6+15+28]:	$S_1(y,x) \rightarrow C_0(y)$	(29)			
			Hyper[7+16+28]:	$S_2(y,x) \rightarrow C_0(y)$	(30)			

Figure 4.1: Example motivating consequence-based calculi

the DL-clauses of (2) and (4) for $0 \leq i \leq n$, this repeated application of DL-clauses constructs a tree-shaped model of depth n and a fanout of two, where nodes at depth i are labelled by B_i and C_i. Forward chaining ensures that reasoning is goal-oriented. However, all nodes with B_i in their label are labelled identically, which reveals a weakness of (hyper)tableau calculi: the constructed models can be large (exponential in our example) and highly redundant; apart from causing problems in practice, this often prevents (hyper)tableau calculi from being worst-case optimal. The reasoning algorithms for the $\mathcal{EL}$ family of DLs [BBL05] avoid this problem by satisfying multiple existential restrictions using the same node, but such an approach is unsound in expressive DLs such as $\mathcal{ALCHIQ}^+$. Moreover, techniques such as *caching* [GN07] or *anywhere blocking* [MSH09] can sometimes constrain model construction, but the effectiveness of these techniques often depends on the order of inference rule applications. In particular, unlike for tableau algorithms for related logics [BDS93, DLNS98], anywhere blocking does not improve the worst-case complexity of the hypertableau calculus for $\mathcal{SHIQ}$ ontologies: it still runs in 2NEXPTIME on $\mathcal{SHIQ}$ [MSH09]. Thus, model size is a key limiting factor for (hyper)tableau-based reasoners.

In contrast, resolution describes models using (universally quantified) clauses that 'summarise' the model. Many resolution variants have been proposed [BG01], each restricting infer-

ences in a specific way, and they were used to develop worst-case optimal decision procedures for a number of decidable fragments of first-order logic [dNSH00, GHS03, GdN99, HMS08]. However, to ensure termination, all such procedures for DLs of which we are aware perform inferences with an atom containing all variables in a clause, and, if there are several such atoms, then a 'deepest' atom is selected. Thus, all resolution-based decision procedures we know of resolve all clauses of (1) with all clauses of (6), and all clauses of (3) with all clauses of (7), and so they eventually derive $2n^2$ clauses of the form (4.32) and (4.33) for $0 \leq i, j < n$.

$$B_i(x) \wedge C_{j+1}(f_{i+1}(x)) \rightarrow C_j(x) \tag{4.32}$$

$$B_i(x) \wedge C_{j+1}(g_{i+1}(x)) \rightarrow C_j(x) \tag{4.33}$$

Of these $2n^2$ clauses, only those with $i = j$ are relevant to proving our goal. If we extend $\mathcal{O}_1$ with additional clauses that contain B_i and C_i, each of these $2n^2$ clauses can participate in further inferences and give rise to more irrelevant clauses. This problem is particularly pronounced when such an extended $\mathcal{O}_1$ is satisfiable since resolution must then produce all such consequences of the ontology in order to prove satisfiability.

4.2 Basics of Consequence-Based Reasoning

Consequence-based calculi combine 'summarisation' of resolution with goal-directed search of (hyper)tableau calculi. Simančík et al. [SMH14] presented a reasoning framework for $\mathcal{ALCI}$ that captures the key elements of the related consequence-based calculi by Baader et al. [BBL05], Kazakov [Kaz09], Ortiz et al. [ORS10], and Simančík et al. [SKH11]. Before extending this framework to $\mathcal{ALCHIQ}^+$ in Chapter 5, we next informally recapitulate the basic notions; however, to make this thesis easier to follow, we use the same notation and terminology as will be introduced in Chapter 5.

Our consequence-based calculus constructs a directed graph $\mathcal{D} = \langle \mathcal{V}, \mathcal{E}, \mathcal{S}, \mathsf{core}, \succ \rangle$ called a *context structure* that consists of the following components:

- $\mathcal{V}$ is a set of vertices where each vertex is called a *context*,
- $\mathcal{E}$ is a set of directed edges where each edge is labelled with a function symbol,
- $\mathcal{S}$ is a function that associates with each vertex v a set of clauses $\mathcal{S}_v$,

- core is a function that associates with each vertex v a (possibly empty) conjunction core_v of *core* atoms, and
- $\succ$ is a function that associates with each vertex v an atom order $\succ_v$.

In the rest of this section, let I be an Herbrand model of $\mathcal{O}_1$; hence, the domain of I consists of ground terms. Instead of representing each domain element (i.e., ground term) of I separately as in (hyper)tableau calculi, $\mathcal{D}$ can represent several terms by a single context v. The set of core atoms core_v associated with each context $v \in \mathcal{V}$ must hold for all ground terms represented by v. Furthermore, the set $\mathcal{S}_v$ of clauses associated with v captures the constraints that must hold for all ground terms that v represents. Partitioning clauses into sets allows us to restrict the inferences between clause sets and thus eliminate certain irrelevant inferences (unlike in resolution where all clauses belong to a single set). Clauses in $\mathcal{S}_v$ are 'relative' to core_v: for each $\Gamma \to \Delta \in \mathcal{S}_v$, we have $\mathcal{O} \models \mathsf{core}_v \wedge \Gamma \to \Delta$. In other words, we do not include core_v in clause bodies since core_v holds implicitly; this convention simplifies the formalisation of the calculus. Finally, the atom order $\succ_v$ associated with v restricts resolution inferences in the presence of disjunctions.

Contexts are connected by directed edges labelled with function symbols. If context u is connected to context v via an f-labelled edge, then context v represents the f-successor of each ground term represented by context u. Conversely, if u and v are *not* connected by an f-edge, then each ground term represented by v is not an f-successor of a ground term represented by u, so no inference between $\mathcal{S}_u$ and $\mathcal{S}_v$ is ever needed.

Consequence-based calculi are not just complete for refutation: they are also guaranteed to derive all consequences of a certain form which is sufficient for checking entailment of query clauses (see Section 2.6). Figure 4.1 demonstrates how such calculi prove entailment for the case of $\mathcal{O}_1 \models B_0(x) \to C_0(x)$. The cores and the clauses are shown above and below, respectively, each context, and clause numbers correspond to the order of derivation. To prove $B_0(x) \to C_0(x)$, the context structure is initialised by introducing context v_0 with core $B_0(x)$ and adding clause (8) to the context. The clause says that B_0 holds for some domain element, and it is analogous to initialising a (hyper)tableau calculus with $B_0(a)$, where a is a fresh constant. The calculus then applies the inference rules (which we will present later in Table 5.1 in Chapter 5) to derive new clauses and/or extend $\mathcal{D}$.

The Hyper rule is the standard hyperresolution rule restricted to a single context. Thus, the

rule derives (9) from (1) and (8), and (10) from (2) and (8); and it derives (11) from (3) and (8), and (12) from (4) and (8). Hyperresolution resolves all body atoms, which makes the resolvent relevant for the context and prevents the derivation of irrelevant clauses such as (4.32) and (4.33).

Context v_0 contains atoms with function symbols f_1 and g_1, and so the Succ rule ensures that the f_1- and g_1-successors of the ground terms represented by v_0 are adequately represented in $\mathcal{D}$. We can control context introduction via a parameter of the calculus called the *expansion strategy*—a function that determines whether to reuse an existing context or introduce a fresh one; if a fresh one is introduced, the strategy also determines how to initialise the core of a context. We discuss different possible strategies in Chapter 5, and defer the precise definitions of their behaviour until then. In the rest of this example, we use the so-called *cautious* strategy for the introduction and reuse of contexts. With this strategy, the Succ rule first fires for function symbol f_1 occurring in clauses (9) and (10); the cautious strategy introduces a fresh context v_1 with core $B_1(x)$, which is analogous to the $\exists$-rule in tableau calculi. Moreover, the Succ rule also adds (15) to v_1, which is analogous to 'pushing' information to the successors using the $\forall$-rule in tableau calculi. The Succ rule next fires for function symbol g_1 occurring in clauses (11) and (12). Since the context structure already contains a context with core $B_1(x)$, the cautious strategy chooses to reuse context v_1, and so the Succ rule adds (16). The Core rule introduces (17) to initialise the context.

Contexts $v_2, \ldots, v_n$ are constructed analogously. Then, (25) is derived by hyperresolving (5) and (24), and (26) is derived by hyperresolving (6), (22), and (25). Clause (26) imposes a constraint on the predecessor context, which is propagated 'backwards' using the Pred rule several times to derive (28). Clause (29) is derived by hyperresolving (6), (15), and (28), and then clause (29) is propagated 'backwards' using the Pred rule to derive (31). Since clauses of v_0 are 'relative' to the core of v_0, clause (31) represents $\mathcal{O}_1 \models B_0(x) \rightarrow C_0(x)$, as required.

Thus, like resolution, consequence-based calculi attempt to 'summarise' models to prevent redundant computation, and, like (hyper)tableau calculi, they attempt to differentiate elements in a model of $\mathcal{O}_1$ to prevent the derivation of consequences such as (4.32) and (4.33). Hence, consequence-based calculi can be seen as a combination of resolution and hypertableau: as in resolution, they describe a model of an ontology by systematically deriving relevant consequences; and as in hypertableau, they avoid drawing certain unnecessary consequences in a goal-directed fashion.

Ontology $\mathcal{O}_2$

$B_0 \sqsubseteq \exists S^-.B_1$	$\rightsquigarrow$	$B_0(x) \to S(f_1(x), x)$	(34)
		$B_0(x) \to B_1(f_1(x))$	(35)
$B_1 \sqsubseteq \exists S.B_2$	$\rightsquigarrow$	$B_1(x) \to S(x, f_2(x))$	(36)
		$B_1(x) \to B_2(f_2(x))$	(37)
$B_1 \sqsubseteq \exists S.B_3$	$\rightsquigarrow$	$B_1(x) \to S(x, f_3(x))$	(38)
		$B_1(x) \to B_3(f_3(x))$	(39)
$B_2 \sqsubseteq B_4$	$\rightsquigarrow$	$B_2(x) \to B_4(x)$	(40)
$B_3 \sqsubseteq B_4$	$\rightsquigarrow$	$B_3(x) \to B_4(x)$	(41)
$B_2 \sqcap B_3 \sqsubseteq \bot$	$\rightsquigarrow$	$B_2(x) \wedge B_3(x) \to \bot$	(42)
$B_1 \sqsubseteq \leq 2.S$	$\rightsquigarrow$	$B_1(x) \wedge S(x, z_1) \wedge S(x, z_2) \wedge S(x, z_3) \to z_1 \approx z_2 \vee z_1 \approx z_3 \vee z_2 \approx z_3$	(43)

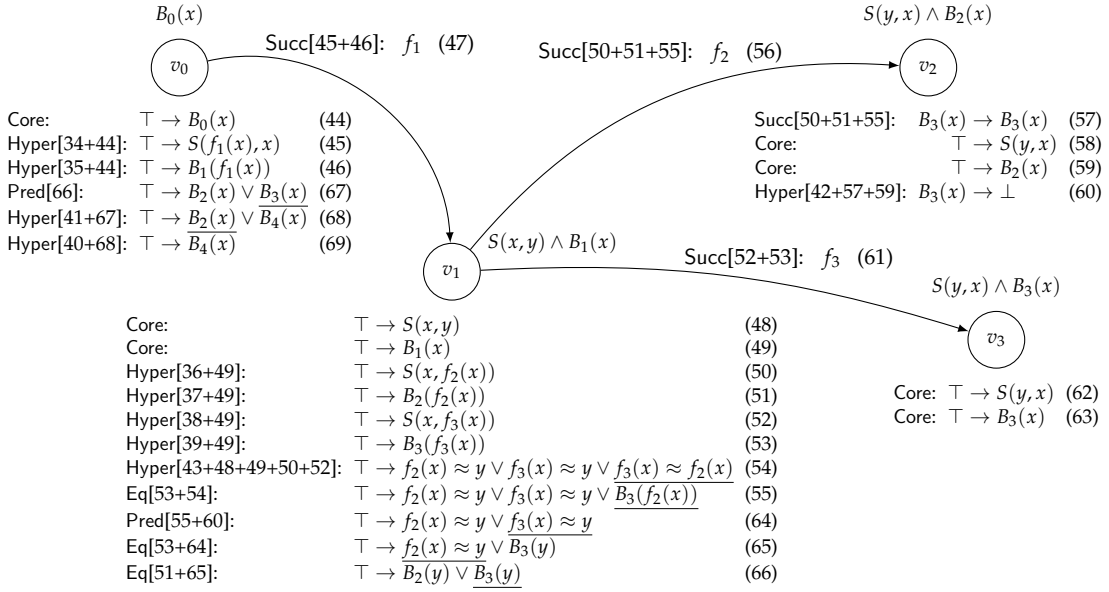


Figure 4.2: Challenges in extending the consequence-based framework to $\mathcal{ALCHIQ}^+$

4.3 Extending the Framework to $\mathcal{ALCHIQ}^+$

In all consequence-based calculi presented thus far, the constraints that the ground terms represented by a context v must satisfy can be represented using standard DL-style axioms. For example, for $\mathcal{ALCI}$, Simančík et al. [SMH14] represented all relevant consequences using DL axioms of the following form, where B_i, B_j, and B_k are atomic concepts, and S_k and S_ℓ are (not necessarily atomic) roles:

$$\bigsqcap B_i \sqsubseteq \bigsqcup B_j \sqcup \bigsqcup \exists S_k.B_k \sqcup \bigsqcup \forall S_\ell.B_\ell \tag{4.70}$$

However, the description logic $\mathcal{ALCHIQ}^+$ provides both cardinality constraints and dis-

junctions, the interplay of which can impose constraints on the model that cannot be represented in $\mathcal{ALCHIQ}^+$. Let $\mathcal{O}_2$ be as in Figure 4.2. To see that $\mathcal{O}_2 \models B_0(x) \rightarrow B_4(x)$ holds, we construct an Herbrand interpretation I using the fresh constant a from $B_0(a)$ by iteratively applying the standard hyperresolution inferences: (34) and (35) derive $S(f_1(a), a)$ and $B_1(f_1(a))$; and (36) and (37) derive $S(f_1(a), f_2(f_1(a)))$ and $B_2(f_2(f_1(a)))$; and (38) and (39) derive $S(f_1(a), f_3(f_1(a)))$ and $B_3(f_3(f_1(a)))$. Due to (40) we derive $B_4(f_2(f_1(a)))$ and due to (41) we derive $B_4(f_3(f_1(a)))$. Finally, from (43) we derive the following clause:

$$f_2(f_1(a)) \approx a \vee f_3(f_1(a)) \approx a \vee f_3(f_1(a)) \approx f_2(f_1(a)) \tag{4.71}$$

Disjunct $f_3(f_1(a)) \approx f_2(f_1(a))$ cannot be satisfied due to (42); but then, regardless of whether we choose to satisfy $f_3(f_1(a)) \approx a$ or $f_2(f_1(a)) \approx a$, we derive $B_4(a)$.

Our calculus must be able to capture constraint (4.71) and its consequences, but standard DL axioms cannot explicitly refer to specific successors and predecessors. Thus, in our calculus we capture consequences using *context clauses*—clauses over terms x, $f_i(x)$, and y, where variable x represents each ground term t in a model I of $\mathcal{O}$ that a context stands for, $f_i(x)$ represents each term $f_i(t)$ in I, and y represents the predecessor of t in I (if one exists). Thus the variable y and the terms $f_i(x)$ provide us with names for the predecessor and the successors, respectively, of each term that x stands for. We can thus capture constraint (4.71) as

$$f_2(x) \approx y \vee f_3(x) \approx y \vee f_3(x) \approx f_2(x). \tag{4.72}$$

Based on this idea, we adapted the inference rules by Simančík et al. [SMH14] to handle context clauses correctly, and we added inference rules that capture the consequences of equality. The resulting set of inference rules is shown in Table 5.1 in Chapter 5.

Figure 4.2 shows how to verify $\mathcal{O}_2 \models B_0(x) \rightarrow B_4(x)$ using our calculus. The maximal literal of each non-Horn clause is underlined. We next discuss the inferences in detail.

We first create context v_0 and initialise it with (44); this ensures that each interpretation represented by the context structure contains a ground term for which B_0 holds. Next, we derive (45) and (46) using hyperresolution. At this point, we could hyperresolve (36) and (46) to obtain $\top \rightarrow S(f_1(x), f_2(f_1(x)))$; however, this could easily lead to non-termination of the calculus

due to increased term nesting. Therefore, we require hyperresolution to map variable x in the DL-clauses to variable x in the context clauses. Because of this restriction, hyperresolution derives only consequences about x within each context.

The Succ rule next handles function symbol f_1 in clauses (45) and (46). To determine which information to propagate to a successor, Definition 2 in Chapter 5 introduces a set $\mathsf{Su}(\mathcal{O}_2)$ of *successor triggers* for the ontology $\mathcal{O}_2$. In our example, DL-clause (43) contains atoms $B_1(x)$ and $S(x, z_i)$ in its body, and z_i can be mapped to a predecessor or a successor of x; thus, a context in which hyperresolution is applied to (43) will be interested in information about its predecessors, which we reflect by adding $B_1(x)$ and $S(x, y)$ to $\mathsf{Su}(\mathcal{O}_2)$. In this example, we use the so-called *eager* strategy (see Chapter 5), so the Succ rule introduces context v_1, and sets its core to $S(x, y)$ and $B_1(x)$. The Core rule initialises the context with (48) and (49).

We next introduce (50)–(53) using hyperresolution, at which point we have sufficient information to apply hyperresolution to (43) to derive (54). Please note how the presence of (48) is crucial for this inference.

We use paramodulation to deal with the equality in clause (54). As is common in resolution-based theorem proving, we order the literals in a clause and apply inferences only to maximal literals; thus, we derive (55).

Clauses (50), (51), and (55) contain function symbol f_2, so the Succ rule introduces context v_2. Due to clause (51), $B_2(x)$ holds for all ground terms that v_2 represents, so we add $B_2(x)$ to core_{v_2}. In contrast, atom $B_3(f_2(x))$ occurs in clause (55) in a disjunction, which means it may (but need not) hold in v_2, and so we add $B_3(x)$ to the body of clause (57). Please note that clause (57) is *not* a tautology in our calculus: the clause essentially says that $B_3(t)$ may be present in a model of $\mathcal{O}_2$ for a ground term t that context v_2 represents; this, in turn, allows us to derive (60) using hyperresolution and thus obtain further constraints that must be satisfied for t in case $B_3(t)$ is indeed present.

Clause (60) essentially says '$B_3(f_2(x))$ should not hold in the predecessor', which the Pred rule propagates to v_1 as clause (64); one can understand this inference as hyperresolution of (55) and (60) while observing that term $f_2(x)$ in context v_1 is represented as variable x in context v_2.

After two paramodulation steps, we derive clause (66), which essentially says 'the predecessor must satisfy $B_2(x)$ or $B_3(x)$'. The set $\mathsf{Pr}(\mathcal{O}_2)$ of *predecessor triggers* for the ontology $\mathcal{O}_2$ (which we will define in Definition 2 in Chapter 5) identifies this as relevant to v_0: DL-clauses (40) and

(41) contain $B_2(x)$ and $B_3(x)$, respectively, in their bodies, which are represented in v_1 as $B_2(y)$ and $B_3(y)$. Hence, $\mathsf{Pr}(\mathcal{O}_2)$ contains $B_2(y)$ and $B_3(y)$, which allows the Pred rule to derive (67).

After two more steps, we finally derive our target clause (69). We could not do this if $B_4(x)$ were maximal in (68); thus, we require all atoms in the head of a query clause to be smallest in the partial order on literals. A similar observation applies to $\mathsf{Pr}(\mathcal{O}_2)$: if $B_3(y)$ were maximal in (65), we would not derive (66) and propagate it to v_0; thus, all atoms in $\mathsf{Pr}(\mathcal{O}_2)$ must also be smallest in the partial order on literals.

Chapter 5

Algorithm

In this chapter, we formally define our consequence-based algorithm for $\mathcal{ALCHIQ}^+$. In Section 5.1, we define our consequence-based calculus and suggest various expansion strategies. In Section 5.2, we describe how the calculus can be used for ontology classification, and present some examples demonstrating various aspects of the calculus. In Section 5.3, we consider the worst-case optimality of the calculus. In Section 5.4, we discuss some of the challenges in designing a calculus for a description logic that supports both cardinality constraints and inverses, as well as the cause of incompleteness in an earlier calculus.

5.1 Definitions

Our calculus manipulates *context clauses*, which are constructed from *context terms* and *context literals* as described in Definition 1. Unlike in general resolution, we restrict context clauses to contain only variables x and y, which have a special meaning in our setting: variable x represents a ground term in an Herbrand model, and y represents the predecessor of x; this naming convention is important for the inference rules of our calculus. This is in contrast to the DL-clauses of an ontology, which can contain variables x and z_i, and where z_i refer to either the predecessor or a successor of x.

For the rest of this chapter, let B denote an abstract concept symbol of type $\mathsf{a} \to \mathsf{p}$, let S denote an abstract role symbol of type $\mathsf{a} \times \mathsf{a} \to \mathsf{p}$, and let f denote an abstract successor function symbol of type $\mathsf{a} \to \mathsf{a}$.

Definition 1 (Context terms, atoms, literals, and clauses)**.** *A* context a-term *is a term of the form* y, x, *or* $f(x)$*; a* context p-term *is a term of the form* $B(y)$, $B(x)$, $B(f(x))$, $S(x,y)$, $S(y,x)$, $S(x,x)$, $S(x,f(x))$, $S(f(x),x)$, *or the constant* true*; and a* context term *is a context* a*-term or a context* p*-term. A* context atom *is an equality of the form* $A \approx \mathsf{true}$ *where* A *is a context* p*-term other than* true*; and a* context literal *is a context atom or a literal of the form* $l \bowtie r$ *where* l *and* r *are context* a*-terms and* $\bowtie \in \{\approx, \not\approx\}$*. A* context clause *is a clause that contains only atoms of the form* $B(x)$, $S(x,x)$, $S(y,x)$, *and* $S(x,y)$ *in the body and only context literals in the head.*

Definition 2 introduces, for a given ontology $\mathcal{O}$, the sets $\mathsf{Su}(\mathcal{O})$ and $\mathsf{Pr}(\mathcal{O})$ that identify the information that must be exchanged between adjacent contexts. Intuitively, $\mathsf{Su}(\mathcal{O})$ contains atoms that are of interest to a successor of a context, and this set is used by the Succ rule. For example, assume that a context u contains a clause containing the atom $B(f(x))$; then, the Succ rule requires that there exists a context v connected to u by an f-labelled edge in which this atom is represented as $B(x)$. In contrast, $\mathsf{Pr}(\mathcal{O})$ contains literals that are of interest to a predecessor of a context, and it is used in the Pred rule. Please note that both $\mathsf{Su}(\mathcal{O})$ and $\mathsf{Pr}(\mathcal{O})$ capture the relevant literals *from the perspective of the successor context*.

Definition 2 (Triggers)**.** *The set* $\mathsf{Su}(\mathcal{O})$ *of* successor triggers *of an ontology* $\mathcal{O}$ *is the smallest set of atoms such that, for each clause* $\Gamma \to \Delta \in \mathcal{O}$*,*

- $B(x) \in \Gamma$ *implies* $B(x) \in \mathsf{Su}(\mathcal{O})$*,*
- $S(x,z_i) \in \Gamma$ *implies* $S(x,y) \in \mathsf{Su}(\mathcal{O})$*, and*
- $S(z_i,x) \in \Gamma$ *implies* $S(y,x) \in \mathsf{Su}(\mathcal{O})$*.*

The set $\mathsf{Pr}(\mathcal{O})$ *of* predecessor triggers *of* $\mathcal{O}$ *is defined as the set of literals*

$$\mathsf{Pr}(\mathcal{O}) = \{\, A\{x \mapsto y,\ y \mapsto x\} \mid A \in \mathsf{Su}(\mathcal{O}) \,\} \cup \{\, B(y) \mid B \text{ occurs in } \mathcal{O} \,\} \cup \{x \approx y\}.$$

Note that all elements of $\mathsf{Pr}(\mathcal{O})$ apart from $x \approx y$ are atoms. As in resolution, we restrict the inferences using a term order $\succ$. Definition 3 specifies the conditions that the order must satisfy. Conditions (i) and (ii) ensure that context a-terms are compared uniformly across contexts; however, context p-terms can be compared in different ways in different contexts. Conditions (i) through (iv) ensure that, if we ground the order by mapping x to a term t and y to the predecessor of t, we obtain a *simplification order* (see Section 2.4). Finally, condition (v) ensures that atoms

that might be propagated to a predecessor of a context via the Pred rule are smallest, which is important for completeness. Please remember that true is a special constant used in the encoding of predicate symbols as function symbols (see Section 2.2).

For a given ontology $\mathcal{O}$, we fix a total, well-founded order $\succcurlyeq$ on function symbols. This is always possible, since the number of function symbols that occur in $\mathcal{O}$ is finite. This order is used in the following definition, which prescribes conditions for any order on context terms.

Definition 3 (Context term order and context literal order)**.** *A* context term order $\succ$ *is an order on context terms satisfying the following conditions:*

(i) for each context p*-term A such that $A \neq \mathsf{true}$, we have $A \succ x \succ y \succ \mathsf{true}$;*

(ii) for all $f, g \in \Sigma^F_{\mathsf{a}}$ with $f \succcurlyeq g$, we have $f(x) \succ g(x)$;

(iii) for all context terms s_1 and s_2 such that $s_1 \succ s_2$, we have $t[s_1]_p \succ t[s_2]_p$ for each context term t and each position p in t;

(iv) for each context term s and each proper position p in s, we have $s \succ s|_p$; and

(v) for each atom $A \approx \mathsf{true} \in \mathsf{Pr}(\mathcal{O})$ and each context term $s \notin \{x, y, \mathsf{true}\}$, we have $A \not\succ s$.

Each context term order is extended to a context literal order*, also written $\succ$, as described in Section 2.3.*

A lexicographic path order (LPO) [BN98] over context a-terms and context p-terms, in which x and y are treated as constants such that $x \succ y \succ \mathsf{true}$, satisfies conditions (i) through (iv). Furthermore, $\mathsf{Pr}(\mathcal{O})$ contains only atoms of the form $B(y)$, $S(x, y)$, and $S(y, x)$, which we can always put as the smallest atoms in $\succ$; thus, condition (v) does not contradict the other conditions. Hence, an LPO that is relaxed for condition (v) satisfies Definition 3, and thus, for any given $\succcurlyeq$, at least one context term order exists. More specifically, we can define the *lexicographic path context term order* $\succ_{lpo}$ to be an order on context terms such that:

(i) for all context terms $s = f(s_1, \ldots, s_m)$ with $m \geq 1$, we have $s \succ_{lpo} x \succ_{lpo} y \succ_{lpo} \mathsf{true}$; and

(ii) for all context terms $s = f(s_1, \ldots, s_m)$ and $t = g(t_1, \ldots, t_n)$, we have $s \succ_{lpo} t$ if and only if $s \approx \mathsf{true} \in \mathsf{Pr}(\mathcal{O})$ implies $t \in \{x, y, \mathsf{true}\}$, and one of the following conditions hold:

(a) $f \succcurlyeq g$ and, for all i with $1 \leq i \leq n$, we have $s \succ_{lpo} t_i$; or

(b) $f = g$ and, for some j with $1 \leq j \leq m$, we have $(s_1, \ldots, s_{j-1}) = (t_1, \ldots, t_{j-1})$, $s_j \succ_{lpo} t_j$, and for all k with $j < k \leq n$, we have $s \succ_{lpo} t_k$; or

(c) $s_j \succeq_{lpo} t$ for some j with $1 \leq j \leq m$.

Apart from orders, effective redundancy elimination techniques are critical to the efficiency of resolution calculi. Definition 4 defines a notion compatible with our setting.

Definition 4 (Redundancy). *A set of clauses U* contains a clause $\Gamma \to \Delta$ up to redundancy, *written* $\Gamma \to \Delta \mathrel{\hat{\in}} U$, *if*

(i) $\{s \approx s',\ s \not\approx s'\} \subseteq \Delta$ *or* $s \approx s \in \Delta$ *for some terms s and s', or*

(ii) $\Gamma' \subseteq \Gamma$ *and* $\Delta' \subseteq \Delta$ *for some clause* $\Gamma' \to \Delta' \in U$.

Intuitively, if U contains $\Gamma \to \Delta$ up to redundancy, then adding $\Gamma \to \Delta$ to U will not modify the constraints that U represents because either $\Gamma \to \Delta$ is a tautology or U contains a stronger clause. Note that tautologies of the form $A \to A$ are *not* redundant in our setting as they are used to initialise contexts; however, whenever our calculus derives a clause $A \to A \vee A'$, the set of clauses will have been initialised with $A \to A$, which makes the former clause redundant by condition (ii) of Definition 4. Moreover, clause heads are subjected to the usual tautology elimination rules; thus, clauses $\Gamma \to \Delta \vee s \approx s$ and $\Gamma \to \Delta \vee s \approx t \vee s \not\approx t$ can be eliminated. Proposition 1 shows that we can remove from U each clause C that is contained in $U \setminus \{C\}$ up to redundancy. The Elim rule from our calculus uses this property to support backward redundancy elimination.

Proposition 1. *Given a set of clauses U and clauses C and C' such that $C \mathrel{\hat{\in}} U \setminus \{C\}$ and $C' \mathrel{\hat{\in}} U$, we have $C' \mathrel{\hat{\in}} U \setminus \{C\}$.*

We are finally ready to formalise the notion of a context structure, as well as a notion of context structure soundness. The latter captures the fact that context clauses from a set $\mathcal{S}_v$ do not contain core_v in their bodies. We shall later show that our inference rules preserve context structure soundness, which essentially proves that all clauses derived by our calculus are indeed conclusions of the input ontology.

Definition 5 (Context structure). *A* context structure *for an ontology $\mathcal{O}$ is defined as a 5-tuple $\mathcal{D} = \langle \mathcal{V}, \mathcal{E}, \mathcal{S}, \mathsf{core}, \succ \rangle$, where*

(i) $\mathcal{V}$ *is a finite set of* contexts,

(ii) $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V} \times \Sigma_{\mathsf{a}}^{F}$ *is a finite set of edges each labelled with a function symbol,*

(iii) $\mathcal{S}$ *is a function that assigns to each context* v *a finite set* $\mathcal{S}_v$ *of context clauses,*

(iv) core *is a function that assigns to each context* v *a conjunction* core_v *of atoms over the* p*-terms from* $\mathsf{Su}(\mathcal{O})$*, and*

(v) $\succ$ *is a function that assigns to each context* v *a context term order* $\succ_v$*.*

A context structure $\mathcal{D}$ *is* sound *for* $\mathcal{O}$ *if the following conditions both hold.*

S1. For each context $v \in \mathcal{V}$ *and each clause* $\Gamma \to \Delta \in \mathcal{S}_v$*, we have* $\mathcal{O} \models \mathsf{core}_v \wedge \Gamma \to \Delta$*.*

S2. For each edge $\langle u, v, f \rangle \in \mathcal{E}$*, we have* $\mathcal{O} \models \mathsf{core}_u \to \mathsf{core}_v\{x \mapsto f(x), y \mapsto x\}$*.*

Observe how the above definition gives semantics to clauses and edges in a context structure. Here we assume that each clause in $\mathcal{S}_v$ implicitly contains core_v in the body. For example, the clause $\top \to A(x) \in S_v$ represents the axiom $\mathsf{core}_v \to A(x)$. An edge $\langle u, v, f_k \rangle$ asserts that each individual of type core_u has an f_k-successor of type core_v. For example, if $\mathsf{core}_v = R(y, x) \wedge B(x)$, then the edge $\langle u, v, f_k \rangle$ represents the axiom $\mathsf{core}_u \to R(x, f_k(x)) \wedge B(f_k(x))$.

Definition 6 introduces the notion of an expansion strategy—a parameter of our calculus that determines when and how to reuse contexts in order to satisfy existential restrictions.

Definition 6 (Expansion strategy)**.** *An* expansion strategy *is a polynomial-time computable function* $\mathsf{strategy}$ *that takes a function symbol* f*, a set of atoms* K*, and a context structure* $\mathcal{D} = \langle \mathcal{V}, \mathcal{E}, \mathcal{S}, \mathsf{core}, \succ \rangle$*. The result of* $\mathsf{strategy}(f, K, \mathcal{D})$ *is a triple* $\langle v, \mathsf{core}', \succ' \rangle$ *with the following components:*

(i) core' *is a subset of* K*;*

(ii) either $v \notin \mathcal{V}$ *is a fresh context not occurring in* $\mathcal{D}$*, or* $v \in \mathcal{V}$ *is an existing context from* $\mathcal{D}$ *such that* $\mathsf{core}_v = \mathsf{core}'$*; and*

(iii) $\succ'$ *is a context term order.*

The chosen expansion strategy is used by the Succ rule of our calculus, as shown in Table 5.1. Note that when a fresh context v is returned by the strategy, the Succ rule ensures that core_v, $\succ_v$, and $\mathcal{S}_v$ are appropriately initialised. We propose the following strategies, where each returns exactly one context per core and each is allowed to return any context term order that satisfies Definition 3.

Table 5.1: Rules of the consequence-based calculus

Core	If	$A \in \mathsf{core}_v$,
		and $\top \to A \notin \mathcal{S}_v$,
	then	add $\top \to A$ to $\mathcal{S}_v$.
Hyper	If	$\bigwedge_{i=1}^{n} A_i \to \Delta \in \mathcal{O}$,
		σ is a substitution such that $\sigma(x) = x$,
		$\Gamma_i \to \Delta_i \vee A_i\sigma \in \mathcal{S}_v$ with $\Delta_i \not\succeq_v A_i\sigma$ for $1 \leq i \leq n$,
		and $\bigwedge_{i=1}^{n} \Gamma_i \to \Delta\sigma \vee \bigvee_{i=1}^{n} \Delta_i \,\hat{\notin}\, \mathcal{S}_v$,
	then	add $\bigwedge_{i=1}^{n} \Gamma_i \to \Delta\sigma \vee \bigvee_{i=1}^{n} \Delta_i$ to $\mathcal{S}_v$.
Eq	If	$\Gamma_1 \to \Delta_1 \vee s_1 \approx t_1 \in \mathcal{S}_v$ with $s_1 \succ_v t_1$ and $\Delta_1 \not\succeq_v s_1 \approx t_1$,
		$\Gamma_2 \to \Delta_2 \vee s_2 \bowtie t_2 \in \mathcal{S}_v$ with $\bowtie \in \{\approx, \not\approx\}$ and $s_2 \succ_v t_2$ and $\Delta_2 \not\succeq_v s_2 \bowtie t_2$,
		p is a position such that $s_2|_p = s_1$ and $s_2|_p$ is not a variable,
		and $\Gamma_1 \wedge \Gamma_2 \to \Delta_1 \vee \Delta_2 \vee s_2[t_1]_p \bowtie t_2 \,\hat{\notin}\, \mathcal{S}_v$,
	then	add $\Gamma_1 \wedge \Gamma_2 \to \Delta_1 \vee \Delta_2 \vee s_2[t_1]_p \bowtie t_2$ to $\mathcal{S}_v$.
Ineq	If	$\Gamma \to \Delta \vee t \not\approx t \in \mathcal{S}_v$,
		and $\Gamma \to \Delta \,\hat{\notin}\, \mathcal{S}_v$,
	then	add $\Gamma \to \Delta$ to $\mathcal{S}_v$.
Factor	If	$\Gamma \to \Delta \vee s \approx t \vee s \approx t' \in \mathcal{S}_v$ with $\Delta \cup \{s \approx t\} \not\succeq_v s \approx t'$ and $s \succ_v t'$,
		and $\Gamma \to \Delta \vee t \not\approx t' \vee s \approx t' \,\hat{\notin}\, \mathcal{S}_v$,
	then	add $\Gamma \to \Delta \vee t \not\approx t' \vee s \approx t'$ to $\mathcal{S}_v$.
Elim	If	$\Gamma \to \Delta \in \mathcal{S}_v$ and
		$\Gamma \to \Delta \,\hat{\in}\, \mathcal{S}_v \setminus \{\Gamma \to \Delta\}$
	then	remove $\Gamma \to \Delta$ from $\mathcal{S}_v$.
Pred	If	$\langle u, v, f \rangle \in \mathcal{E}$,
		$\bigwedge_{i=1}^{m} A_i \to \bigvee_{i=m+1}^{m+n} L_i \in \mathcal{S}_v$,
		$\Gamma_i \to \Delta_i \vee A_i\sigma \in \mathcal{S}_u$ with $\Delta_i \not\succeq_u A_i\sigma$ for $1 \leq i \leq m$,
		$L_i \in \mathsf{Pr}(\mathcal{O})$ for each $m+1 \leq i \leq m+n$,
		and $\bigwedge_{i=1}^{m} \Gamma_i \to \bigvee_{i=1}^{m} \Delta_i \vee \bigvee_{i=m+1}^{m+n} L_i\sigma \,\hat{\notin}\, \mathcal{S}_u$,
	then	add $\bigwedge_{i=1}^{m} \Gamma_i \to \bigvee_{i=1}^{m} \Delta_i \vee \bigvee_{i=m+1}^{m+n} L_i\sigma$ to $\mathcal{S}_u$;
	where	$\sigma = \{x \mapsto f(x), y \mapsto x\}$.
Succ	If	$\Gamma \to \Delta \vee A \in \mathcal{S}_u$ where $\Delta \not\succeq_u A$ and A contains $f(x)$, and
		no edge $\langle u, v, f \rangle \in \mathcal{E}$ exists such that $A' \to A' \,\hat{\in}\, \mathcal{S}_v$ for each $A' \in K_2 \setminus \mathsf{core}_v$,
	then	let $\langle v, \mathsf{core}', \succ' \rangle := \mathsf{strategy}(f, K_1, \mathcal{D})$;
		if $v \in \mathcal{V}$, then let $\succ_v := \succ_v \cap \succ'$, and
		otherwise let $\mathcal{V} := \mathcal{V} \cup \{v\}$, $\mathsf{core}_v := \mathsf{core}'$, $\succ_v := \succ'$, and $\mathcal{S}_v := \emptyset$;
		add the edge $\langle u, v, f \rangle$ to $\mathcal{E}$; and
		add $A' \to A'$ to $\mathcal{S}_v$ for each $A' \in K_2 \setminus \mathsf{core}_v$;
	where	$\sigma = \{x \mapsto f(x), y \mapsto x\}$,
		$K_1 = \{\, A' \in \mathsf{Su}(\mathcal{O}) \mid \top \to A'\sigma \in \mathcal{S}_u \,\}$, and
		$K_2 = \{\, A' \in \mathsf{Su}(\mathcal{O}) \mid \Gamma' \to \Delta' \vee A'\sigma \in \mathcal{S}_u$ and $\Delta' \not\succeq_u A'\sigma \,\}$.

- The *trivial* strategy introduces just one 'trivial' context $v_\top$ with the empty core (i.e., we have $\mathsf{core}_{v_\top} = \top$). All consequences are then kept in a single set, and so our calculus becomes similar to the known resolution-based decision procedures.
- The *eager* strategy returns for each K_1 the context v_{K_1} with core K_1. The 'kind' of ground terms that v_{K_1} represents is then very specific, and so the set $\mathcal{S}_{v_{K_1}}$ is likely to be smaller but the number of contexts can be exponential.
- The *cautious* strategy examines the function symbol f: if there exists an abstract concept symbol B such that both (i) f occurs in $\mathcal{O}$ in exactly one atom of the form $B(f(x))$ and (ii) $B(x) \in K_1$, then the result is the context $v_{B(x)}$ with core $B(x)$; otherwise, the result is the 'trivial' context $v_\top$ with the empty core. Condition (i) is trivially satisfied for DL-clauses obtained by transforming an $\mathcal{ALCHIQ}^+$ ontology in DL-style syntax as described in Section 2.6. Context $v_{B(x)}$ is then less constrained, but the number of contexts is at most linear.

Although we do not discuss it here, strategies that return multiple contexts with the same core are also possible. Simančík et al. [SMH14] discuss extensively the differences between the eager and the cautious strategy and their relative merits; although their discussion deals with $\mathcal{ALCI}$ only, their conclusions apply to $\mathcal{ALCHIQ}^+$ and $\mathcal{SRIQ}$ as well.

We are now ready to state the properties of soundness and completeness. Proofs of our results are provided in Chapter 6.

Theorem 1 (Soundness). *For an arbitrary expansion strategy, applying an inference rule from Table 5.1 to an ontology $\mathcal{O}$ and a context structure $\mathcal{D}$ that is sound for $\mathcal{O}$ produces a context structure that is sound for $\mathcal{O}$.*

Theorem 2 (Completeness). *Let $\mathcal{O}$ be an ontology, and let $\mathcal{D} = \langle \mathcal{V}, \mathcal{E}, \mathcal{S}, \mathsf{core}, \succ \rangle$ be a context structure such that no inference rule from Table 5.1 is applicable to $\mathcal{O}$ and $\mathcal{D}$. Then, $\Gamma_Q \to \Delta_Q \hat{\in} \mathcal{S}_q$ holds for each query clause $\Gamma_Q \to \Delta_Q$ and each context $q \in \mathcal{V}$ such that conditions C1 through C3 are satisfied.*

C1. $\mathcal{O} \models \Gamma_Q \to \Delta_Q$.

C2. For each context atom $A \approx \mathsf{true} \in \Delta_Q$ and each context atom $A' \approx \mathsf{true}$ not containing variable y such that $A \succ_q A'$, we have $A' \approx \mathsf{true} \in \Delta_Q$.

C3. For each $A \in \Gamma_Q$, we have $\Gamma_Q \to A \hat{\in} \mathcal{S}_q$.

The completeness theorem essentially says that, given an ontology $\mathcal{O}$ and a context structure $\mathcal{D}$ that is saturated under the rules of the calculus, for each context v, we can read off entailed clauses of the form $\Gamma \to \Delta$ from the set $\mathcal{S}_v$ provided that $\Gamma \to \Delta$ is a query clause, the context literal order $\succ_v$ when restricted to atoms not containing variable y is such that the atoms in Δ are the smallest, and the set $\mathcal{S}_v$ is appropriately initialized.

5.2 Instantiating the Context Structure for Ontology Classification

Conditions C2 and C3 of Theorem 2 can be satisfied by appropriately initialising the corresponding context. Hence, Theorems 1 and 2 show that the following algorithm is sound and complete for deciding $\mathcal{O} \models \Gamma_Q \to \Delta_Q$.

A1. Create an empty context structure $\mathcal{D}$ and select an expansion strategy.

A2. Introduce a context q into $\mathcal{D}$; set $\mathsf{core}_q = \Gamma_Q$; for each $A \in \Gamma_Q$, add $\top \to A$ to $\mathcal{S}_q$ to satisfy condition C3; and initialise $\succ_q$ in a way that satisfies condition C2.

A3. Apply the inference rules from Table 5.1 to $\mathcal{D}$ and $\mathcal{O}$ until no inference rule is applicable.

A4. $\mathcal{O} \models \Gamma_Q \to \Delta_Q$ holds if and only if $\Gamma_Q \to \Delta_Q \hat{\in} \mathcal{S}_q$.

Observe that the algorithm is don't-care nondeterministic: firstly, the conclusion of an inference rule is added to $\mathcal{S}_q$ only if $\mathcal{S}_q$ does not contain a strengthening of that conclusion, and secondly, the order of inference rule applications is not predetermined. Hence, the resulting context structure $\mathcal{D}$ is not unique. Nonetheless, this is don't-care nondeterminism; any order of inference rule applications suffices for soundness and completeness. However, imposing an order on the inference rules may be necessary for ensuring pay-as-you-go behaviour on fragments of $\mathcal{ALCHIQ}^+$, and this is discussed in Section 5.3.

If we wish to decide entailment of several query clauses, we can simply apply step A2 to each query clause independently—that is, we introduce and initialise a context q_i for each distinct body of a query clause. We then apply the inference rules to such a context structure, after which we read off entailment of each query clause as in step A4 from the appropriate $\mathcal{S}_{q_i}$. Hence we obtain the following algorithm to classify an ontology $\mathcal{O}$.

B1. Create an empty context structure $\mathcal{D}$ and select an expansion strategy.

B2. Introduce a context $q_\top$ into $\mathcal{D}$, and set $\mathsf{core}_{q_\top} = \top$.

Ontology $\mathcal{O}_3$

$\mathsf{Giraffe} \sqsubseteq \mathsf{Animal}$	$\rightsquigarrow$	$\mathsf{Giraffe}(x) \rightarrow \mathsf{Animal}(x)$	(1)
		$\mathsf{Animal}(x) \rightarrow \mathsf{Vegetarian}(x) \vee \mathsf{Aux}_1(x)$	(2)
$\mathsf{Animal} \sqcap \forall \mathsf{eats}.\neg\mathsf{Animal} \sqsubseteq \mathsf{Vegetarian}$	$\rightsquigarrow$	$\mathsf{Aux}_1(x) \rightarrow \mathsf{Animal}(f_1(x))$	(3)
		$\mathsf{Aux}_1(x) \rightarrow \mathsf{eats}(x, f_1(x))$	(4)
$\mathsf{Giraffe} \sqsubseteq \forall \mathsf{eats}.\mathsf{Leaf}$	$\rightsquigarrow$	$\mathsf{Giraffe}(x) \wedge \mathsf{eats}(x, z_1) \rightarrow \mathsf{Leaf}(z_1)$	(5)
$\mathsf{Leaf} \sqsubseteq \exists \mathsf{partOf}.\mathsf{Plant}$	$\rightsquigarrow$	$\mathsf{Leaf}(x) \rightarrow \mathsf{partOf}(x, f_2(x))$	(6)
		$\mathsf{Leaf}(x) \rightarrow \mathsf{Plant}(f_2(x))$	(7)
$\mathsf{Animal} \sqcap \exists \mathsf{partOf}.\mathsf{Plant} \sqsubseteq \bot$	$\rightsquigarrow$	$\mathsf{partOf}(z_1, x) \wedge \mathsf{Plant}(x) \rightarrow \mathsf{Aux}_2(z_1)$	(8)
		$\mathsf{Animal}(x) \wedge \mathsf{Aux}_2(x) \rightarrow \bot$	(9)

$\mathsf{Su}(\mathcal{O}_3) = \{\mathsf{Giraffe}(x), \mathsf{Animal}(x), \mathsf{Aux}_1(x), \mathsf{eats}(x,y), \mathsf{Leaf}(x), \mathsf{partOf}(y,x), \mathsf{Plant}(x), \mathsf{Aux}_2(x)\}$
$\mathsf{Pr}(\mathcal{O}_3) = \{\mathsf{Giraffe}(y), \mathsf{Animal}(y), \mathsf{Aux}_1(y), \mathsf{eats}(y,x), \mathsf{Leaf}(y), \mathsf{partOf}(x,y), \mathsf{Plant}(y), \mathsf{Aux}_2(y), \mathsf{Vegetarian}(y), x \approx y\}$

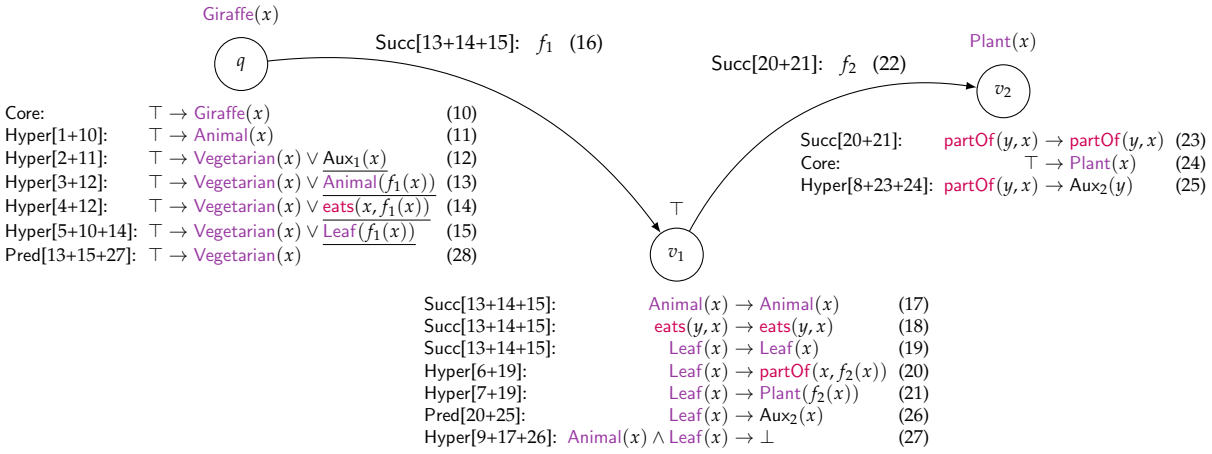


Figure 5.1: Example demonstrating the necessity of the order condition C2 of Theorem 2

B3. For each abstract concept symbol B_i in $\mathcal{O}$, introduce a context q_i into $\mathcal{D}$, set $\mathsf{core}_{q_i} = B_i(x)$, and add $\top \rightarrow B_i(x)$ to $\mathcal{S}_{q_i}$.

B4. For each context q added in steps B2 and B3, initialise $\succ_q$ to be such that, for each abstract concept symbol B' in $\mathcal{O}$, and each context atom $A' \approx \mathsf{true}$ not containing variable y we have $B'(x) \not\succ_q A'$.

B5. Apply the inference rules from Table 5.1 to $\mathcal{D}$ and $\mathcal{O}$ until no inference rule is applicable.

B6. $\mathcal{O} \models \top \rightarrow \bot$ holds if and only if $\top \rightarrow \bot \hat{\in} \mathcal{S}_{q_\top}$.

B7. For all pairs of abstract concept symbols B_i and B_j in $\mathcal{O}$, we have that $\mathcal{O} \models B_i(x) \rightarrow B_j(x)$ holds if and only if $B_i(x) \rightarrow B_j(x) \hat{\in} \mathcal{S}_{q_i}$.

We now consider an example demonstrating the necessity of the order condition C2 of Theorem 2. Consider the example ontology $\mathcal{O}_3$ which is shown in Figure 5.1; by inspection we

can see that $\mathcal{O}_3 \models \mathsf{Giraffe}(x) \rightarrow \mathsf{Vegetarian}(x)$. Our goal is to prove this using algorithm A1–A4. For this example, we will use the cautious strategy. Step A2 of the algorithm requires that $\succ_q$ is initialised in a way that satisfies condition C2. Since $\mathsf{Aux}_1(x)$ is a context atom not containing variable y, condition C2 requires the relative ordering of $\mathsf{Vegetarian}(x)$ and $\mathsf{Aux}_1(x)$ to be such that $\mathsf{Vegetarian}(x) \not\succ_q \mathsf{Aux}_1(x)$. For this example, we have chosen $\succ_q$ to be such that $\mathsf{Aux}_1(x) \succ_q \mathsf{Vegetarian}(x)$.

The triggers of $\mathcal{O}_3$ and the saturated context structure are also shown in Figure 5.1. Each context is shown as a node, with the core of the context shown above the node and the clauses belonging to the context shown below the node, and the maximal literals in the head of each non-Horn clause are underlined. In particular, in clause (12), the maximal literal is $\mathsf{Aux}_1(x)$. Clause numbers correspond to the order of derivation. The calculus derives clause (28) in context q with $\mathsf{core}_q = \{\mathsf{Giraffe}(x)\}$, which indeed proves that $\mathcal{O}_3 \models \mathsf{Giraffe}(x) \rightarrow \mathsf{Vegetarian}(x)$.

If, however, in our example $\succ_q$ had instead been initialised in a way that does not satisfy condition C2 of the completeness theorem, then it is not guaranteed that our calculus can prove $\mathcal{O}_3 \models \mathsf{Giraffe}(x) \rightarrow \mathsf{Vegetarian}(x)$. For example, if $\succ_q$ was instead chosen to be such that $\mathsf{Vegetarian}(x) \succ_q \mathsf{Aux}_1(x)$, then starting from the empty context structure, after initialisation with clause (10), the Hyper rule derives (11) and (12) as before, but $\mathsf{Vegetarian}(x)$ would instead be the maximal literal in clause (12). But then no more clauses would be derived because $\mathsf{Vegetarian}(x)$ does not occur in the body of any clause in $\mathcal{O}_3$, and hence failing to satisfy condition C2 of Theorem 2 can lead to incompleteness.

Next, we consider a simple example that demonstrates reasoning with both equality and disjunction using our calculus. Consider the ontology $\mathcal{O}_4$ which is shown in Figure 5.2. Our goal is to prove that $\mathcal{O}_4 \models A(x) \rightarrow D(x)$ using algorithm A1–A4. For this example, we use will the cautious strategy. The triggers of $\mathcal{O}_4$ and the resulting context structure are also shown in Figure 5.2. Our goal is proved because, in the saturated context structure, clause (50) has been derived in context q with $\mathsf{core}_q = \{A(x)\}$.

In our final example, we consider the ontology $\mathcal{O}_5$ which is shown in Figure 5.3. Our goal is to prove that $\mathcal{O}_5 \models \mathsf{Vegetarian}(x) \rightarrow \mathsf{Person}(x)$ using algorithm A1–A4 and the eager strategy. Please note that multiple applications of the Pred, Eq and Ineq rules have each been performed in one step in order to save space in the figure. Observe how multiple f_i-successors of the context q

Ontology $\mathcal{O}_4$

$A \sqsubseteq \exists R.B$	$\rightsquigarrow$	$A(x) \to R(x, f_1(x))$	(29)
		$A(x) \to B(f_1(x))$	(30)
$A \sqsubseteq \exists R.C$	$\rightsquigarrow$	$A(x) \to R(x, f_2(x))$	(31)
		$A(x) \to C(f_2(x))$	(32)
$B \sqcap C \sqsubseteq \bot$	$\rightsquigarrow$	$B(x) \wedge C(x) \to \bot$	(33)
$\geqslant 2\, R.\top \sqsubseteq D$	$\rightsquigarrow$	$R(x, z_1) \wedge R(x, z_2) \to D(x) \vee z_1 \approx z_2$	(34)

$$\mathsf{Su}(\mathcal{O}_4) = \{A(x), B(x), C(x), R(x, y)\}$$
$$\mathsf{Pr}(\mathcal{O}_4) = \{A(y), B(y), C(y), R(y, x), D(y), x \approx y\}$$

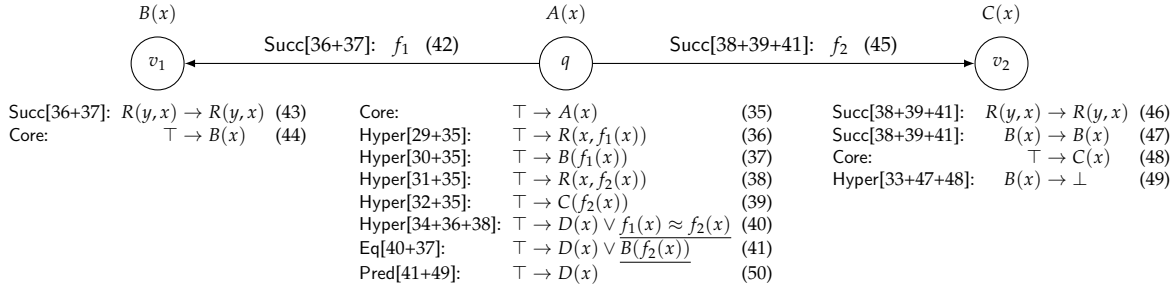


Figure 5.2: Example demonstrating reasoning with equality and disjunction

are represented by the same context. Our goal is indeed proved, since clause (72) is derived in context q with $\mathsf{core}_q = \{\mathsf{Vegetarian}(x)\}$.

5.3 Worst-Case Optimality and Pay-As-You-Go Behaviour

In this section, we consider the worst-case optimality of the calculus, and discuss the behaviour of the calculus on Horn ontologies. Propositions 2 and 3 show that our calculus is worst-case optimal for both $\mathcal{ALCHIQ}^+$ and $\mathcal{ELH}$. Proofs are given in Chapter 6.

Proposition 2. *For each expansion strategy that introduces at most exponentially many contexts, algorithm A1–A4 runs in worst-case exponential time.*

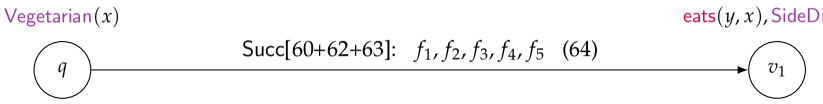
This is worst-case optimal since concept satisfiability with respect to a general TBox in $\mathcal{ALC}$ is EXPTIME-hard [BCMN+10].

Proposition 3. *For $\mathcal{ELH}$ ontologies and queries of the form $B_1(x) \to B_2(x)$, algorithm A1–A4 runs in polynomial time with either the cautious or the eager strategy; and with the cautious strategy and the* Hyper *rule applied eagerly, the inferences in step A3 correspond directly to the inferences of the $\mathcal{ELH}$ calculus by Baader et al. [BBL05].*

Ontology $\mathcal{O}_5$

$\mathsf{Vegetarian} \sqsubseteq \mathsf{Animal} \sqcap \forall \mathsf{eats}.\mathsf{SideDish}$	$\rightsquigarrow$	$\mathsf{Vegetarian}(x) \to \mathsf{Animal}(x)$	(51)	
		$\mathsf{Vegetarian}(x) \wedge \mathsf{eats}(x, z_1) \to \mathsf{SideDish}(z_1)$	(52)	
$\mathsf{Animal} \sqsubseteq \geqslant 5\, \mathsf{eats}$	$\rightsquigarrow$	$\mathsf{Animal}(x) \to \mathsf{eats}(x, f_i(x))$	(53)	for $1 \leq i \leq 5$
		$\mathsf{Animal}(x) \to f_i(x) \not\approx f_j(x)$	(54)	for $1 \leq i < j \leq 5$
$\mathsf{Meat} \sqcap \mathsf{SideDish} \sqsubseteq \bot$	$\rightsquigarrow$	$\mathsf{Meat}(x) \wedge \mathsf{SideDish}(x) \to \bot$	(55)	
$\geqslant 5\, \mathsf{eats}.\neg \mathsf{Meat} \sqsubseteq \mathsf{HealthyPerson}$	$\rightsquigarrow$	$\bigwedge_{1 \leq i \leq 5} \mathsf{eats}(x, z_i) \to \mathsf{HealthyPerson}(x) \vee \bigvee_{1 \leq i < j \leq 5} z_i \approx z_j \vee \bigvee_{1 \leq i \leq 5} \mathsf{Meat}(z_i)$	(56)	
$\mathsf{HealthyPerson} \sqsubseteq \mathsf{Person}$	$\rightsquigarrow$	$\mathsf{HealthyPerson}(x) \to \mathsf{Person}(x)$	(57)	

$$\mathsf{Su}(\mathcal{O}_5) = \{\mathsf{Vegetarian}(x), \mathsf{eats}(x,y), \mathsf{Animal}(x), \mathsf{Meat}(x), \mathsf{SideDish}(x), \mathsf{HealthyPerson}(x)\}$$
$$\mathsf{Pr}(\mathcal{O}_5) = \{\mathsf{Vegetarian}(y), \mathsf{eats}(y,x), \mathsf{Animal}(y), \mathsf{Meat}(y), \mathsf{SideDish}(y), \mathsf{HealthyPerson}(y), \mathsf{Person}(y), x \approx y\}$$



Core:	$\top \to \mathsf{Vegetarian}(x)$	(58)	Succ[60+62+63]:	$\mathsf{Meat}(x) \to \mathsf{Meat}(x)$	(65)
Hyper[51+58]:	$\top \to \mathsf{Animal}(x)$	(59)	Core:	$\top \to \mathsf{eats}(y, x)$	(66)
Hyper[53+59]:	$\top \to \mathsf{eats}(x, f_i(x))$	(60)	Core:	$\top \to \mathsf{SideDish}(x)$	(67)
Hyper[54+59]:	$\top \to f_i(x) \not\approx f_j(x)$	(61)	Hyper[55+65+67]:	$\mathsf{Meat}(x) \to \bot$	(68)
Hyper[52+58+60]:	$\top \to \mathsf{SideDish}(f_i(x))$	(62)			
Hyper[56+60]:	$\top \to \mathsf{HealthyPerson}(x) \vee \bigvee_{1 \leq i < j \leq 5} f_i(x) \approx f_j(x) \vee \bigvee_{1 \leq i \leq 5} \mathsf{Meat}(f_i(x))$	(63)			
Pred[63+68]:	$\top \to \mathsf{HealthyPerson}(x) \vee \bigvee_{1 \leq i < j \leq 5} f_i(x) \approx f_j(x)$	(69)			
Eq[61+69]:	$\top \to \mathsf{HealthyPerson}(x) \vee \bigvee_{1 \leq i \leq 5} f_i(x) \not\approx f_i(x)$	(70)			
Ineq[70]:	$\top \to \mathsf{HealthyPerson}(x)$	(71)			
Hyper[57+71]:	$\top \to \mathsf{Person}(x)$	(72)			

Figure 5.3: Extended example demonstrating the workings of the calculus

Ontology $\mathcal{O}_6$

$B \sqsubseteq \neg C$	$\rightsquigarrow$	$B(x) \wedge C(x) \rightarrow \bot$	(73)
$\top \sqsubseteq \exists R.C$	$\rightsquigarrow$	$\top \rightarrow R(x, f_1(x))$	(74)
		$\top \rightarrow C(f_1(x))$	(75)
$\top \sqsubseteq \leqslant 1\, R^-$	$\rightsquigarrow$	$R(z_1, x) \wedge R(z_2, x) \rightarrow z_1 \approx z_1$	(76)

$$\mathsf{Su}(\mathcal{O}_6) = \{B(x), C(x), R(y, x)\}$$
$$\mathsf{Pr}(\mathcal{O}_6) = \{B(y), C(y), R(x, y), x \approx y\}$$

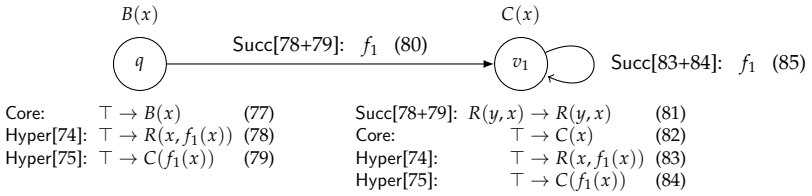


Figure 5.4: Example $\mathcal{ALCFI}$ ontology where B only has an infinite model

Note that, even for an $\mathcal{ELH}$ ontology, when the trivial strategy is used (i.e., reusing just one context) the algorithm may derive clauses with arbitrarily large bodies, and hence potentially run in exponential time.

We now briefly discuss the behaviour of algorithm A1–A4 on Horn-$\mathcal{ALCHIQ}^+$ (i.e., where each DL-clause contains at most one literal in the head). All inference rules in Table 5.1 are Horn-preserving, so given a Horn ontology the algorithm only derives Horn context clauses. Furthermore, when using the eager strategy, the algorithm only derives context clauses with empty bodies; therefore, there are at most polynomially many context clauses in each context, but the number of contexts may be exponential. Note that, when using a strategy other than the eager strategy, the algorithm can derive Horn context clauses with non-empty bodies.

5.4 Difficulties in Combining Cardinality Constraints and Inverses

As previously discussed, it is well known that designing a calculus for a description logic that combines both cardinality constraints and inverse roles is challenging. In this section, we briefly review some of these difficulties, before examining the incompleteness of a previous calculus for Horn-$\mathcal{SHIQ}$.

On the model-theoretic side, the combination of cardinality constraints and inverses causes a logic to lose the finite-model property. For example, consider the ontology $\mathcal{O}_6$ which is shown

in Figure 5.4. The concept B only has an infinite model with respect to ontology $\mathcal{O}_6$. The context structure constructed by our calculus using the cautious strategy to test the satisfiability of B is also shown in the figure.

Furthermore, on the proof-theoretic side, this combination of constructors makes tableau and hypertableau calculi more complex by requiring anywhere pairwise blocking to ensure termination [HST00a], whereas in our calculus, termination is comparatively straightforward.

Finally, this combination of constructors also causes a resolution-based decision procedure to be more complex because, unlike in decision procedures for related logics [GdN99, SH03], it is necessary to consider clauses containing terms of depth two [Mot06]. Our calculus captures this by using y to represent the predecessor of x and $f_i(x)$ to represent the f_i-successor of x.

The interactions between cardinality constraints and inverse roles can be complex, even if the logic does not allow for disjunction. For example, consider the ontology $\mathcal{O}_7$ which is shown in Figure 5.5. This example demonstrates that the calculus presented by Kazakov [Kaz09] for Horn-$\mathcal{SHIQ}$ is incomplete. Specifically, the concept A is unsatisfiable, as proven by our calculus in (153), but the calculus of Kazakov is unable to derive this. The interaction that causes the Kazakov calculus to be incomplete is that, in any minimal model of $\mathcal{O}_7$, both R_2 and R_3 must be enforced between the same pairs of individuals, and so must R_1 and R_3, and so must R_2 and R_4. However, the Kazakov calculus does not capture that R_1 and R_4 must be enforced between the same pairs of individuals and thus does not derive $A \sqsubseteq \bot$ from $A \sqsubseteq \exists R_1.B$, $A \sqsubseteq \exists R_4.B$ and $B \sqcap E \sqsubseteq \bot$.

One possible fix for the Kazakov calculus would be to capture the set of roles that must be enforced between pairs of individuals in the model. This is the approach taken by Ortiz et al. [ORS10] in their algorithm for reasoning in Horn-$\mathcal{SROIQ}$.

However, this possible treatment of roles does not easily extend to a logic that additionally supports disjunction, and our calculus is the first consequence-based procedure of which we are aware that correctly handles the combination of disjunctions, cardinality constraints, and inverse roles.

Ontology $\mathcal{O}_7$

$A \sqsubseteq \exists R_1.B$	$\rightsquigarrow$	$A(x) \to R_1(x, f_1(x))$	(86)
		$A(x) \to B(f_1(x))$	(87)
$B \sqsubseteq \exists R_2^-.C$	$\rightsquigarrow$	$B(x) \to R_2(f_2(x), x)$	(88)
		$B(x) \to C(f_2(x))$	(89)
$C \sqsubseteq \exists R_3.D$	$\rightsquigarrow$	$C(x) \to R_3(x, f_3(x))$	(90)
		$C(x) \to D(f_3(x))$	(91)
$A \sqsubseteq \exists R_4.E$	$\rightsquigarrow$	$A(x) \to R_4(x, f_4(x))$	(92)
		$A(x) \to E(f_4(x))$	(93)
$R_1 \sqsubseteq S_{13}$	$\rightsquigarrow$	$R_1(x, z_1) \to S_{13}(x, z_1)$	(94)
$R_3 \sqsubseteq S_{13}$	$\rightsquigarrow$	$R_3(x, z_1) \to S_{13}(x, z_1)$	(95)
$R_2 \sqsubseteq S_{23}$	$\rightsquigarrow$	$R_2(x, z_1) \to S_{23}(x, z_1)$	(96)
$R_3 \sqsubseteq S_{23}$	$\rightsquigarrow$	$R_3(x, z_1) \to S_{23}(x, z_1)$	(97)
$R_2 \sqsubseteq S_{24}$	$\rightsquigarrow$	$R_2(x, z_1) \to S_{24}(x, z_1)$	(98)
$R_4 \sqsubseteq S_{24}$	$\rightsquigarrow$	$R_4(x, z_1) \to S_{24}(x, z_1)$	(99)
$B \sqcap E \sqsubseteq \bot$	$\rightsquigarrow$	$B(x) \wedge E(x) \to \bot$	(100)
$\top \sqsubseteq \leqslant 1\, S_{13}^-$	$\rightsquigarrow$	$S_{13}(z_1, x) \wedge S_{13}(z_2, x) \to z_1 \approx z_2$	(101)
$\top \sqsubseteq \leqslant 1\, S_{23}$	$\rightsquigarrow$	$S_{23}(x, z_1) \wedge S_{23}(x, z_2) \to z_1 \approx z_2$	(102)
$\top \sqsubseteq \leqslant 1\, S_{24}$	$\rightsquigarrow$	$S_{24}(x, z_1) \wedge S_{24}(x, z_2) \to z_1 \approx z_2$	(103)

$\mathsf{Su}(\mathcal{O}_7) = \{A(x), B(x), C(x), R_1(x,y), R_3(x,y), R_2(x,y), R_4(x,y), E(x), S_{13}(y,x), S_{23}(x,y), S_{24}(x,y)\}$
$\mathsf{Pr}(\mathcal{O}_7) = \{A(y), B(y), C(y), R_1(y,x), R_3(y,x), R_2(y,x), R_4(y,x), E(y), S_{13}(x,y), S_{23}(y,x), S_{24}(y,x), D(y), x \approx y\}$

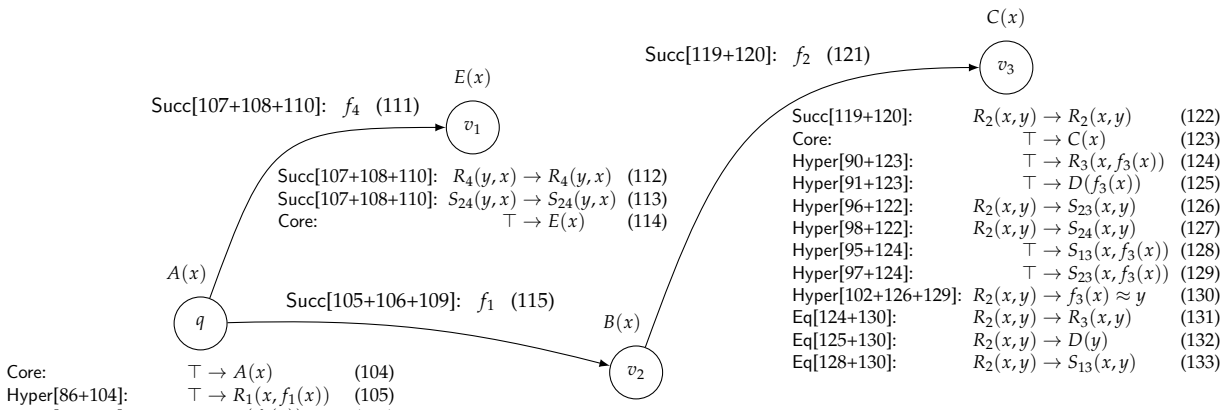


Figure 5.5: Example ontology demonstrating the incompleteness of a previous approach and the context structure constructed by our calculus using the cautious strategy

Chapter 6

Soundness & Completeness

In this chapter, we show that our calculus is both sound and complete. Furthermore, we prove that the calculus is worst-case optimal for both $\mathcal{ALCHIQ}^+$ and $\mathcal{ELH}$.

In Section 6.1, we show that the calculus is sound. In Section 6.2, we present an outline of the completeness proof, followed by the full proof of completeness in Section 6.3. Finally, in Section 6.4, we prove worst-case optimality.

6.1 Proof of the Soundness Theorem

In this section, we show that our calculus is sound, as stated in Theorem 1. The proof is analogous to the soundness proof of ordered superposition [NR95], but we also show that all inferences produce context clauses. We recall the soundness theorem:

Theorem 1 (Soundness)**.** *For an arbitrary expansion strategy, applying an inference rule from Table 5.1 to an ontology $\mathcal{O}$ and a context structure $\mathcal{D}$ that is sound for $\mathcal{O}$ produces a context structure that is sound for $\mathcal{O}$.*

Proof. Let $\mathcal{O}$ be an ontology and let $\mathcal{D} = \langle \mathcal{V}, \mathcal{E}, \mathcal{S}, \mathsf{core}, \succ \rangle$ be a context structure that is sound for $\mathcal{O}$. For each inference rule from Table 5.1 applied to clauses, contexts, and edges as shown in the table, we show that the clause produced by the inference rule is a context clause that satisfies condition S1 of Definition 5; moreover, for the Succ rule, we show in addition that the edge introduced by the inference rule satisfies condition S2. Inference rules other than Hyper, Eq, and Pred obviously produce context clauses. Moreover, our proofs rely on the soundness of

hyperresolution: for arbitrary formulas ω, ϕ_i, ψ_i, and γ_i, $1 \leq i \leq n$, we have

$$\{\bigwedge_{i=1}^{n} \phi_i \to \omega\} \cup \bigcup_{1 \leq i \leq n} \{\gamma_i \to \psi_i \vee \phi_i\} \models \bigwedge_{i=1}^{n} \gamma_i \to \omega \vee \bigvee_{i=1}^{n} \psi_i. \tag{6.1}$$

(Core rule) For each $A \in \mathsf{core}_v$, we clearly have $\mathcal{O} \models \mathsf{core}_v \to A$.

(Hyper rule) Property (6.2) clearly holds since $\bigwedge_{i=1}^{n} A_i \to \Delta \in \mathcal{O}$, and property (6.3) holds since context structure $\mathcal{D}$ is sound for $\mathcal{O}$. But then, property (6.2) implies property (6.4), which, together with (6.1) and (6.3), implies (6.5), as required for condition S1.

$$\mathcal{O} \models \bigwedge_{i=1}^{n} A_i \to \Delta \tag{6.2}$$

$$\mathcal{O} \models \mathsf{core}_v \wedge \Gamma_i \to \Delta_i \vee A_i\sigma \qquad \text{for } 1 \leq i \leq n \tag{6.3}$$

$$\mathcal{O} \models \bigwedge_{i=1}^{n} A_i\sigma \to \Delta\sigma \tag{6.4}$$

$$\mathcal{O} \models \mathsf{core}_v \wedge \bigwedge_{i=1}^{n} \Gamma_i \to \Delta\sigma \vee \bigvee_{i=1}^{n} \Delta_i \tag{6.5}$$

Finally, substitution σ satisfies $\sigma(x) = x$, all premises are context clauses, and $\mathcal{O}$ contains only DL-clauses; thus, the inference rule can only match an atom $S(x,x)$, $S(x,z_i)$ or $S(z_i,x)$ in an ontology clause to atoms $S(y,x)$, $S(x,y)$, $S(x,x)$, $S(f(x),x)$, or $S(x,f(x))$ in the context clause, and so $\sigma(z_i)$ is either y or $f(x)$. Consequently, (6.5) is a context clause.

(Eq rule) Since $\mathcal{D}$ is sound for $\mathcal{O}$, properties (6.6) and (6.7) hold. Moreover, the clause in (6.8) is a logical consequence of the clauses in (6.6) and (6.7), so property (6.8) holds, as required.

$$\mathcal{O} \models \mathsf{core}_v \wedge \Gamma_1 \to \Delta_1 \vee s_1 \approx t_1 \tag{6.6}$$

$$\mathcal{O} \models \mathsf{core}_v \wedge \Gamma_2 \to \Delta_2 \vee s_2 \bowtie t_2 \tag{6.7}$$

$$\mathcal{O} \models \mathsf{core}_v \wedge \Gamma_1 \wedge \Gamma_2 \to \Delta_1 \vee \Delta_2 \vee s_2[t_1]_p \bowtie t_2 \tag{6.8}$$

Finally, the inference rule requires that $s_2|_p = s_1$ and that $s_2|_p$ is not a variable, so therefore s_1 is not a variable either. Consequently, term s_1 is always of the form $f(x)$, term t_1 is of the form $g(x)$, x, or y, and term s_2 is of the form $f(x)$,

$B(f(x))$, $S(x, f(x))$, or $S(f(x), x)$; thus, $s_2[t_1]_p$ is a context term, and so the result is a context clause.

(Ineq rule) Since $\mathcal{D}$ is sound for $\mathcal{O}$, we have $\mathcal{O} \models \mathsf{core}_v \wedge \Gamma \to \Delta \vee t \not\approx t$; but then, we clearly have $\mathcal{O} \models \mathsf{core}_v \wedge \Gamma \to \Delta$, as required.

(Factor rule) Since $\mathcal{D}$ is sound for $\mathcal{O}$, property (6.9) holds. Moreover, the clause in (6.10) is a logical consequence of the clause in (6.9), so property (6.10) holds, as required.

$$\mathcal{O} \models \mathsf{core}_v \wedge \Gamma \to \Delta \vee s \approx t \vee s \approx t' \tag{6.9}$$

$$\mathcal{O} \models \mathsf{core}_v \wedge \Gamma \to \Delta \vee t \not\approx t' \vee s \approx t' \tag{6.10}$$

(Elim rule) The resulting context structure contains a subset of the clauses from $\mathcal{D}$, so it is clearly sound for $\mathcal{O}$.

(Pred rule) Since $\mathcal{D}$ is sound for $\mathcal{O}$, properties (6.11) through (6.13) hold. Now the clause in (6.14) is an instance of the clause in (6.11), so property (6.14) holds. But then, by (6.1), properties (6.12) and (6.14) imply property (6.15). Finally, properties (6.13) and (6.15) imply property (6.16).

$$\mathcal{O} \models \mathsf{core}_v \wedge \textstyle\bigwedge_{i=1}^{m} A_i \to \bigvee_{i=m+1}^{m+n} L_i \tag{6.11}$$

$$\mathcal{O} \models \mathsf{core}_u \wedge \Gamma_i \to \Delta_i \vee A_i\sigma \qquad \text{for } 1 \leq i \leq m \tag{6.12}$$

$$\mathcal{O} \models \mathsf{core}_u \to \mathsf{core}_v\sigma \tag{6.13}$$

$$\mathcal{O} \models \mathsf{core}_v\sigma \wedge \textstyle\bigwedge_{i=1}^{m} A_i\sigma \to \bigvee_{i=m+1}^{m+n} L_i\sigma \tag{6.14}$$

$$\mathcal{O} \models \mathsf{core}_v\sigma \wedge \mathsf{core}_u \wedge \textstyle\bigwedge_{i=1}^{m} \Gamma_i \to \bigvee_{i=1}^{m} \Delta_i \vee \bigvee_{i=m+1}^{m+n} L_i\sigma \tag{6.15}$$

$$\mathcal{O} \models \mathsf{core}_u \wedge \textstyle\bigwedge_{i=1}^{m} \Gamma_i \to \bigvee_{i=1}^{m} \Delta_i \vee \bigvee_{i=m+1}^{m+n} L_i\sigma \tag{6.16}$$

For each $m + 1 \leq i \leq m + n$, we have $L_i \in \mathsf{Pr}(\mathcal{O})$, so L_i is of the form $B(y)$, $S(x, y)$, $S(y, x)$, or $x \approx y$; but then, $\sigma = \{x \mapsto f(x), y \mapsto x\}$ ensures that $L_i\sigma$ is of the form $B(x)$, $S(f(x), x)$, $S(x, f(x))$, or $f(x) \approx x$ and so $L_i\sigma$ is a context atom, as required.

(Succ rule) For each clause $A' \to A'$ added to $\mathcal{S}_v$, we clearly have $\mathcal{O} \models \mathsf{core}_v \wedge A' \to A'$, as required for condition S1 of Definition 5. Moreover, assume that the inference rule adds an edge $\langle u, v, f_k \rangle$ to $\mathcal{E}$. Then, the definition of K_1 ensures $\top \to A'\sigma \in \mathcal{S}_u$

for each $A' \in K_1$. Moreover, $\mathcal{D}$ is sound for $\mathcal{O}$, so (6.17) holds for context u, and it implies (6.18).

$$\mathcal{O} \models \mathsf{core}_u \to A'\sigma \qquad \text{for each } A' \in K_1 \tag{6.17}$$

$$\mathcal{O} \models \mathsf{core}_u \to K_1\sigma \tag{6.18}$$

$$\mathcal{O} \models \mathsf{core}_u \to \mathsf{core}_v\sigma \tag{6.19}$$

But then, Definition 6 ensures $\mathsf{core}_v \subseteq K_1$, so property (6.18) holds, as required for condition S2 of Definition 5. □

6.2 An Outline of the Completeness Proof

To prove Theorem 2, we fix an ontology $\mathcal{O}$, a context structure $\mathcal{D}$, a query clause $\Gamma_Q \to \Delta_Q$, and a context q such that conditions C2 and C3 of Theorem 2 and $\Gamma_Q \to \Delta_Q \notin \mathcal{S}_q$ are all satisfied, and we construct an Herbrand interpretation that satisfies $\mathcal{O}$ but refutes $\Gamma_Q \to \Delta_Q$. We reuse techniques from equational theorem proving [NR95] and represent this interpretation by a *rewrite system* R—a finite set of rules of the form $l \Rightarrow r$. Intuitively, such a rule says that any two terms of the form $f_1(\dots f_n(l)\dots)$ and $f_1(\dots f_n(r)\dots)$ with $n \geq 0$ are equal, and that we can prove this equality in one step by rewriting (i.e., replacing) l with r. Rewrite system R induces an Herbrand equality interpretation R^* that contains each $l \approx r$ for which the equality between l and r can be verified using a finite number of such rewrite steps. The universe of R^* consists of a- and p-terms constructed using the symbols in Σ^F_{a} and Σ^F_{p}, and a special constant c.

We obtain R by unfolding the context structure $\mathcal{D}$ starting from context q: we map each a-term $t \in \mathrm{HU}_{\mathsf{a}}$ to a context X_t in $\mathcal{D}$, and we use the clauses in $\mathcal{S}_{X_t}$ to construct a model fragment R_t—the part of R that satisfies the DL-clauses of $\mathcal{O}$ when x is mapped to t. The key issue is to ensure compatibility between adjacent model fragments: when moving from a *predecessor* term t' to a *successor* term $t = f(t')$, we must ensure that adding R_t to $R_{t'}$ does not affect the truth of the DL-clauses of $\mathcal{O}$ at term t'; in other words, the model fragment constructed at t must respect the choices made at t'. We represent these choices by a ground clause $\Gamma_t \to \Delta_t$: conjunction Γ_t contains atoms that are 'inherited' from t' and so must hold at t, and disjunction Δ_t contains atoms that must not hold at t because t' relies on their absence.

The model fragment construction takes as parameters a term t, a context $v = X_t$, and a clause $\Gamma_t \to \Delta_t$. Let N_t be the set of ground clauses obtained from $\mathcal{S}_v$ by mapping x to t and y to the predecessor of t (if it exists), and whose body is contained in Γ_t. Moreover, let Su_t and Pr_t be obtained from $\mathsf{Su}(\mathcal{O})$ and $\mathsf{Pr}(\mathcal{O})$ by mapping x to t and y to the predecessor of t if it exists; thus, Su_t contains the ground atoms of interest to the successors of t, and Pr_t contains the ground atoms of interest to the predecessor of t. The model fragment for t can be constructed if conditions L1 through L3 hold:

L1. $\Gamma_t \to \Delta_t \hat{\notin} N_t$.

L2. If $t = c$, then $\Delta_t = \Delta_Q$; and if $t \neq c$, then we have $\{t \approx t'\} \subseteq \Delta_t \subseteq \mathsf{Pr}_t$ where t' is the predecessor of t.

L3. For each $A \in \Gamma_t$, we have $\Gamma_t \to A \hat{\in} N_t$.

The construction produces a rewrite system R_t such that

F1. $R_t^* \models N_t$, and

F2. $R_t^* \not\models \Gamma_t \to \Delta_t$—that is, all of Γ_t, but none of Δ_t hold in R_t^*, and so the model fragment at t is compatible with the model fragment at the predecessor of t.

We construct rewrite system R_t by adapting the techniques from paramodulation-based theorem proving. First, we order all clauses in N_t into a sequence $C^i = \Gamma^i \to \Delta^i \vee L^i$, $1 \leq i \leq n$, that is compatible with the context term order $\succ_v$ in a particular way. Next, we initialise R_t to $\emptyset$, and then we examine each clause C^i in this sequence; if C^i does not hold in the model constructed thus far, we make the clause true by adding L^i to R_t. To prove condition F1, we assume for the sake of a contradiction that a clause C^i with smallest i exists such that $R_t^* \not\models C^i$, and we show that an application of the Eq, Ineq, or Factor rule to C^i necessarily produces a clause C^j such that $R_t^* \not\models C^j$ and $j < i$. Conditions L1 through L3 allow us to satisfy condition F2. Due to condition L2 and condition (v) of Definition 3, we can order the clauses in the sequence such that each clause C^i capable of producing an atom from Δ_t comes before any other clause in the sequence; and then we use condition L1 to show that no such clause actually exists. Moreover, condition L3 ensures that all atoms in Γ_t are actually produced in R_t^*.

To obtain R, we inductively unfold $\mathcal{D}$, and at each step we apply the model fragment construction to the appropriate parameters. For the base case, we map constant c to context $X_c = q$,

and we define $\Gamma_c = \Gamma_Q$ and $\Delta_c = \Delta_Q$; then, conditions L1 and L2 hold by definition, and condition L3 holds by condition C3 of Theorem 2. For the induction step, we assume that we have already mapped some term t' to a context $u = X_{t'}$, and we consider the term $t = f(t')$ for each $f \in \Sigma^F_{\mathsf{a}}$.

- If t does not occur in an atom in $R_{t'}$, we let $R_t = \{t \Rightarrow c\}$ and thus make t equal to c. Term t is thus interpreted in exactly the same way as c, so we stop the unfolding.
- If $R_{t'}$ contains a rule $t \Rightarrow s$, then t and s are equal, and so we interpret t exactly as s; hence, we stop the unfolding.
- In all other cases, the Succ rule ensures that $\mathcal{D}$ contains an edge $\langle u, v, f\rangle$ such that context v satisfies all preconditions of the inference rule, so we define $X_t = v$. Furthermore, we let $\Gamma_t = R^*_{t'} \cap \mathsf{Su}_t$ be the set of atoms that hold at t' and are relevant to t , and we let $\Delta_t = \mathsf{Pr}_t \setminus R^*_{t'}$ be the set of atoms that do not hold at t' and are relevant to t. We finally show that such Γ_t and Δ_t satisfy condition L1: otherwise, the Pred rule derives a clause in $N_{t'}$ that is not true in $R^*_{t'}$.

After processing all relevant terms, we let R be the union of all R_t from the above construction. To show that R^* satisfies $\mathcal{O}$, we consider a DL-clause $\Gamma \to \Delta \in \mathcal{O}$ and a substitution τ that makes the clause ground. Without loss of generality we can assume that τ is irreducible by R—that is, it does not contain terms that can we rewritten using the rules in R. Since each model fragment satisfies condition F2, we can evaluate $\Gamma\tau \to \Delta\tau$ in $R^*_{\tau(x)}$ instead of R^*. Moreover, we show that $R^*_{\tau(x)} \models \Gamma\tau \to \Delta\tau$ holds: if that were not the case, the Hyper rule derives a clause in $N_{\tau(x)}$ that violates condition F1. Finally, we show that the same holds for the query clause $\Gamma_Q \to \Delta_Q$, which completes our proof.

6.3 Proof of the Completeness Theorem

In this section, we show that our calculus is complete, as stated in Theorem 2:

Theorem 2 (Completeness)**.** *Let $\mathcal{O}$ be an ontology, and let $\mathcal{D} = \langle \mathcal{V}, \mathcal{E}, \mathcal{S}, \mathsf{core}, \succ\rangle$ be a context structure such that no inference rule from Table 5.1 is applicable to $\mathcal{O}$ and $\mathcal{D}$. Then, $\Gamma_Q \to \Delta_Q \hat{\in} \mathcal{S}_q$ holds for each query clause $\Gamma_Q \to \Delta_Q$ and each context $q \in \mathcal{V}$ such that conditions C1 through C3 are satisfied.*

C1. $\mathcal{O} \models \Gamma_Q \to \Delta_Q$.

C2. *For each context atom* $A \approx \mathsf{true} \in \Delta_Q$ *and each context atom* $A' \approx \mathsf{true}$ *not containing variable* y *such that* $A \succ_q A'$, *we have* $A' \approx \mathsf{true} \in \Delta_Q$.

C3. *For each* $A \in \Gamma_Q$, *we have* $\Gamma_Q \to A \hat{\in} \mathcal{S}_q$.

In this section, we fix an ontology $\mathcal{O}$, a context structure $\mathcal{D} = \langle \mathcal{V}, \mathcal{E}, \mathcal{S}, \mathsf{core}, \succ \rangle$, a context $q \in \mathcal{V}$, and a query clause $\Gamma_Q \to \Delta_Q$ such that conditions C2 and C3 of Theorem 2 are satisfied, and we show the contrapositive of condition C1: if $\Gamma_Q \to \Delta_Q \hat{\notin} \mathcal{S}_q$, then $\mathcal{O} \not\models \Gamma_Q \to \Delta_Q$. To this end, we construct a rewrite system R such that the induced Herbrand model R^* satisfies all clauses in $\mathcal{O}$, but not $\Gamma_Q \to \Delta_Q$. We construct the model using a distinguished constant c, the function symbols of sort a, and the symbols of sort p.

Let t be a term. If t is of the form $t = f(s)$, then s is the *predecessor* of t, and t is a *successor* of s; by these definitions, a constant does not have a predecessor. The a-*neighbourhood* of t is the following set of a-terms: t, $f(t)$ for each $f \in \Sigma^F_{\mathsf{a}}$, and the predecessor t' of t if it exists. The p-*neighbourhood* of t is the following set of p-terms: $B(t)$, $S(t,t)$, $S(t,f(t))$, $S(f(t),t)$, and $B(f(t))$, and, if t has the predecessor t', also $S(t',t)$, $S(t,t')$, and $B(t')$, for each $f \in \Sigma^F_{\mathsf{a}}$, each unary symbol $B \in \Sigma^F_{\mathsf{p}}$, and each binary symbol $S \in \Sigma^F_{\mathsf{p}}$. The *neighbourhood* of t is the set containing the a-neighbourhood of t and the p-neighbourhood of t. Let σ_t be the substitution such that $\sigma_t(x) = t$ and, if t has the predecessor t', then $\sigma_t(y) = t'$. Finally, for each term t, we define sets of atoms Pr_t and Su_t as follows:

$$\mathsf{Su}_t = \{\, A\sigma_t \mid A \in \mathsf{Su}(\mathcal{O}) \text{ and } A\sigma_t \text{ is ground} \,\} \tag{6.20}$$

$$\mathsf{Pr}_t = \{\, L\sigma_t \mid L \in \mathsf{Pr}(\mathcal{O}) \text{ and } L\sigma_t \text{ is ground} \,\} \tag{6.21}$$

$$\mathsf{Ref}_t = \{\, S(t,t) \mid S \in \Sigma^F_{\mathsf{p}} \text{ is a binary symbol} \,\} \tag{6.22}$$

6.3.1 Constructing a Model Fragment

In this section, we show how, given a term t, we can generate a part of the model of $\mathcal{O}$ that covers the neighbourhood of t. In the rest of Section 6.3.1, we fix the following parameters to the model fragment generation process:

- t is a ground a-term,
- v is a context in $\mathcal{D}$,

- Γ_t is a conjunction of atoms, and
- Δ_t is a disjunction of atoms.

Let N_t be the set of ground clauses obtained from $\mathcal{S}_v$ as follows:

$$N_t = \{\, \Gamma\sigma_t \to \Delta\sigma_t \mid \Gamma \to \Delta \in \mathcal{S}_v, \text{ both } \Gamma\sigma_t \text{ and } \Delta\sigma_t \text{ are ground, and } \Gamma\sigma_t \subseteq \Gamma_t \,\}$$

We assume that the following conditions hold.

L1. $\Gamma_t \to \Delta_t \mathrel{\hat{\notin}} N_t$.

L2. If $t = c$, then $\Delta_t = \Delta_Q$; and if $t \neq c$, then we have $\{t \approx t'\} \subseteq \Delta_t \subseteq \mathsf{Pr}_t$ where t' is the predecessor of t.

L3. For each $A \in \Gamma_t$, we have $\Gamma_t \to A \mathrel{\hat{\in}} N_t$.

We next construct a rewrite system R_t such that $R_t^* \models N_t$ and $R_t^* \not\models \Gamma_t \to \Delta_t$ holds. Throughout Section 6.3.1, we treat the terms in the a-neighbourhood of t as if they were constants. Thus, even though the rewrite system R will contain terms t and $f(t)$, we will not consider terms with further nesting.

6.3.1.1 Grounding the Context Order

To construct R_t, we need an order on the terms in the neighbourhood of t that is compatible with $\succ_v$. To this end, let $>_t$ be a total, strict, simplification order on the set of ground terms constructed using the a-neighbourhood of t and the symbols of the predicate sort p that satisfies the following conditions for all context terms s_1 and s_2 such that $s_1\sigma_t$ and $s_2\sigma_t$ are both ground, and where t' is the predecessor of t if it exists.

O1. $s_1 \succ_v s_2$ implies $s_1\sigma_t >_t s_2\sigma_t$.

O2. $s_1\sigma_t \approx \mathsf{true} \in \Delta_t$ and $s_2\sigma_t \notin \{t, t', \mathsf{true}\}$ and $s_2\sigma_t \approx \mathsf{true} \notin \Delta_t$ imply $s_2\sigma_t >_t s_1\sigma_t$.

To see that order $\succ_v$ on (nonground) context terms can be grounded in a way that is compatible with these definitions, note that condition O2 can be rewritten as follows:

$$s_1\sigma_t \approx \mathsf{true} \in \Delta_t \text{ and } s_2\sigma_t \notin \{t, t', \mathsf{true}\} \text{ and } s_1\sigma_t \geq_t s_2\sigma_t \text{ imply } s_2\sigma_t \approx \mathsf{true} \in \Delta_t.$$

Now, for $t = c$ (and hence $v = q$), condition C2 of Theorem 2 ensures that such $>_t$ exists; and for $t \neq c$, condition (v) of Definition 3 ensures this as well.

Moreover, since in this section we treat all a-terms as constants, we can make the p-terms of the form $B(t')$, $S(t', t)$, and $S(t, t')$ smaller than other a- and p-terms (i.e., we do not need to worry about defining the order on the predecessor of t' or on the ancestors of $f(t)$). Thus, at least one such order exists, so in the rest of this section we fix an arbitrary such order $>_t$. We extend $>_t$ to ground literals (also written $>_t$) by associating each $s \not\approx t$ with the multiset $\{s, s, t, t\}$ and each $s \approx t$ with the multiset $\{s, t\}$, and then comparing the result using the multiset extension of the term order (as defined in Chapter 2). Due to condition (i) of Definition 3, we have

$$A \approx \mathsf{true} >_t t \not\approx t >_t t \approx t >_t t \not\approx t' >_t t \approx t' >_t t' \not\approx t' >_t t' \approx t' >_t \mathsf{true} \approx \mathsf{true} \tag{6.23}$$

for each context atom A of the form $B(t)$, $B(t')$, $S(t, t')$, $S(t', t)$, and $S(t, t)$. Finally, we further extend $>_t$ to disjunctions of ground literals (also written $>_t$) by identifying each disjunction $\bigvee_{i=1}^{n} L_i$ with the multiset $\{L_1, \ldots, L_n\}$ and then comparing the result using the multiset extension of the literal order.

6.3.1.2 Constructing the Rewrite System R_t

We arrange all clauses in N_t into a sequence $C^1, \ldots, C^n$. Since the body of each C^i is a subset of Γ_t, no C^i can contain $\bot$ in its head since that would contradict condition L1; thus, we can assume that each C^i is of the form $C^i = \Gamma^i \to \Delta^i \vee L^i$ where $L^i >_t \Delta^i$, literal L^i is of the form $L^i = l^i \bowtie r^i$ with $\bowtie \in \{\approx, \not\approx\}$, and $l^i \geq_t r^i$. For the rest of Section 6.3.1, we reserve C^i, Γ^i, Δ^i, L^i, l^i, and r^i for referring to the (parts of) the clauses in this sequence. Finally, we assume that, for all $1 \leq i < j \leq n$, we have $\Delta^j \vee L^j \geq_t \Delta^i \vee L^i$.

Definition 7 (Model fragment)**.** *We define the sequence $R_t^0, \ldots, R_t^n$ of rewrite systems by setting $R_t^0 = \emptyset$ and defining each R_t^i with $1 \leq i \leq n$ inductively as follows:*

- *$R_t^i = R_t^{i-1} \cup \{l^i \Rightarrow r^i\}$ if L^i is of the form $l^i \approx r^i$ such that*

  R1. $(R_t^{i-1})^* \not\models \Delta^i \vee l^i \approx r^i$,

  R2. $l^i >_t r^i$,

  R3. l^i *is irreducible by* R_t^{i-1}*, and*

R4. $s \approx r^i \notin (R_t^{i-1})^*$ *for each* $l^i \approx s \in \Delta^i$*; and*

- $R_t^i = R_t^{i-1}$ *in all other cases.*

Finally, let $R_t = R_t^n$*; we call* R_t *the* model fragment for t, v, Γ_t, and Δ_t.

Each clause $C^i = \Gamma^i \to \Delta^i \vee l^i \approx r^i$ *that satisfies the first condition in the above construction is called* generative*, and the clause is said to* generate *the rule* $l^i \Rightarrow r^i$ *in* R_t.

6.3.1.3 The Properties of the Model Fragment R_t

Lemma 1. *The rewrite system* R_t *is Church-Rosser.*

Proof. To prove that R_t is Church-Rosser, it is sufficient to show that R_t is both terminating and left-reduced. To see that R_t is terminating, simply note that, for each rule $l \Rightarrow r \in R_t$, condition R2 ensures $l >_t r$, and that $>_t$ is a simplification order.

To see that R_t is left-reduced, consider an arbitrary rule $l^i \Rightarrow r^i \in R_t$ that is added to R_t in step i of the clause sequence. By condition R3, $l^i \Rightarrow r^i$ is irreducible by R_t^{i-1}. Now consider an arbitrary rule $l^j \Rightarrow r^j \in R_t$ that is added to R_t at any step j of the construction where $j > i$. The definition of the clause order implies $l^j \approx r^j \geq_t l^i \approx r^i$; since $l^j >_t r^j$ and $l^i >_t r^i$ by condition R2, by the definition of the literal order we have $l^j \geq_t l^i$. Since $l^i \Rightarrow r^i \in R_t^{j-1}$, condition R3 for j ensures $l^i \neq l^j$, and so we have $l^j >_t l^i$; consequently, l^j is not a subterm of l^i, and thus l^i is irreducible by $R_t^j \setminus \{l^i \Rightarrow r^i\}$. □

Lemma 2. *For each* $1 \leq i \leq n$ *and each* $l \not\approx r \in \Delta^i \vee L^i$*, we have that* $(R_t^{i-1})^* \models l \approx r$ *if and only if* $R_t^* \models l \approx r$.

Proof. Consider an arbitrary clause $C^i = \Gamma^i \to \Delta^i \vee L^i$ and an arbitrary inequality $l \not\approx r$ such that $l \not\approx r \in \Delta^i \vee L^i$. If $l \approx r \in (R_t^{i-1})^*$, then $R_t^{i-1} \subseteq R_t$ implies $l \approx r \in R_t^*$, and so we have $R_t^* \models l \approx r$, as required. Now assume that $l \approx r \notin (R_t^{i-1})^*$. Let l' and r' be the normal forms of l and r, respectively, w.r.t. R_t^{i-1}. Now consider an arbitrary j with $i \leq j \leq n$ such that $l^j \Rightarrow r^j$ is generated by C^j. The definitions of the model construction ensure $l^j \approx r^j \geq_t L^i >_t \Delta^i$. Now if $l \not\approx r = L^i$, then $l^j \approx r^j >_t l \not\approx r$ holds because the two literals are different; otherwise, we have $l \not\approx r \in \Delta^i$, and so $l^j \approx r^j >_t l \not\approx r$ holds as well. By the definition of the literal order, we then have $l^j >_t l \geq_t l'$ and $l^j >_t r \geq_t r'$; since $>_t$ is a simplification order, l^j is a subterm of neither l' nor r'. Thus, l' and

r' are the normal forms of l and r, respectively, w.r.t. R_t^j, and so we have $l' \approx r' \notin (R_t^j)^*$; but then, we have $l \approx r \notin (R_t^j)^*$, as required. □

Lemma 3. *For each generative clause* $\Gamma^i \to \Delta^i \vee l^i \approx r^i$, *we have* $R_t^* \not\models \Delta^i$.

Proof. Let $C^i = \Gamma^i \to \Delta^i \vee l^i \approx r^i$ be an arbitrary generative clause and let $L \in \Delta^i$ be an arbitrary literal; condition R1 ensures that $(R_t^{i-1})^* \not\models L$. We next show that $R_t^* \not\models L$.

Assume that L is of the form $l \not\approx r$. Since $l \not\approx r \in \Delta^i \vee l^i \approx r^i$, by Lemma 2 we have $R_t^* \not\models L$, as required.

Assume that L is of the form $l \approx r$ with $l >_t r$. We show by induction that, for each j with $i \leq j \leq n$, we have $(R_t^j)^* \not\models L$. To this end, we assume that $(R_t^{j-1})^* \not\models L$. If C^j is not generative, then $R_t^j = R_t^{j-1}$, and so $(R_t^j)^* \not\models L$. The only remaining possibility is that C^j is generative, for which we consider the following cases.

- $l^j = l$. We have the following two subcases.
  - $j = i$, and so $l = l^i = l^j$. Condition R4 ensures $r \approx r^i \notin (R_t^{i-1})^*$. Let r' and r'' be the normal forms of r and r^i, respectively, w.r.t. R_t^{i-1}; we have $r' \approx r'' \notin (R_t^{i-1})^*$. Moreover, $l >_t r \geq_t r'$ and $l >_t r^i \geq_t r''$ hold; since $>_t$ is a simplification order, l is a subterm of neither r' nor r''; therefore, r' and r'' are the normal forms of r and r^i, respectively, w.r.t. R_t^i, and therefore $r' \approx r'' \notin (R_t^i)^*$. Finally, since $l \Rightarrow r^i \in R_t^i$, term r'' is the normal form of l w.r.t. R_t^i, and so $l \approx r \notin (R_t^i)^*$.
  - $j > i$. But then, $l^j \approx r^j \geq_t l^i \approx r^i >_t l \approx r$ implies $l^j = l^i = l$. Furthermore, C^i is generative, so we have $l^i \Rightarrow r^i \in R_t^{j-1}$. But then, l^j is not irreducible by R_t^{j-1}, which contradicts condition R3.
- $l^j >_t l$. Let l' and r' be the normal forms of l and r, respectively, w.r.t. R_t^{j-1}. Then, we have $l^j >_t l \geq_t l'$ and $l^j >_t r \geq_t r'$; since $>_t$ is a simplification order, l^j is a subterm of neither l' nor r'. Thus, l' and r' are the normal forms of l and r, respectively, w.r.t. R_t^j, and so $l' \approx r' \notin (R_t^j)^*$; hence, $l \approx r \notin (R_t^j)^*$ holds. □

Lemma 4. *Let* $\Gamma \to \Delta$ *be a clause such that* $\Gamma \to \Delta \hat{\in} N_t$. *Then* $R_t^* \models \Delta$ *holds if there exists* i *with* $1 \leq i \leq n+1$ *such that*

(i) for each $1 \leq j < i$, *we have* $R_t^* \models \Delta^j \vee L^j$, *and*

(ii) if $i \leq n$ (i.e., i is an index of a clause from N_t), then $\Delta^i \vee L^i >_t \Delta$.

Proof. Assume that $\Gamma \to \Delta \mathrel{\hat{\in}} N_t$ holds. If $\Gamma \to \Delta$ satisfies condition (i) of Definition 4, then we clearly have $R_t^* \models \Delta$. Assume that $\Gamma \to \Delta$ satisfies condition (ii) of Definition 4 due to some clause $\Gamma^j \to \Delta^j \vee L^j \in N_t$ such that $\Gamma^j \subseteq \Gamma$ and $\Delta^j \cup \{L^j\} \subseteq \Delta$ hold; the latter clearly implies $\Delta \geq_t \Delta^j \vee L^j$. Let i be an integer satisfying the lemma assumption. If $i = n + 1$, then we clearly have $j < i$; otherwise, $\Delta^i \vee L^i >_t \Delta$ implies $\Delta^i \vee L^i >_t \Delta^j \vee L^j$, and so we also have $j < i$. But then, by the lemma assumption we have $R_t^* \models \Delta^j \vee L^j$, which implies $R_t^* \models \Delta$, as required. □

Lemma 5. *For each clause $\Gamma \to \Delta$ such that $\Gamma \to \Delta \mathrel{\hat{\in}} \mathcal{S}_v$ and $\Gamma\sigma_t \subseteq \Gamma_t$ hold, we have $\Gamma\sigma_t \to \Delta\sigma_t \mathrel{\hat{\in}} N_t$.*

Proof. Assume that $\Gamma \to \Delta \mathrel{\hat{\in}} \mathcal{S}_v$ holds. If $\Gamma \to \Delta$ satisfies condition (i) of Definition 4, then terms s and s' exist such that $s \approx s \in \Delta$ or $\{s \approx s',\ s \not\approx s'\} \subseteq \Delta$; but then we have that $s\sigma_t \approx s\sigma_t \in \Delta\sigma_t$ or $\{s\sigma_t \approx s'\sigma_t,\ s\sigma_t \not\approx s'\sigma_t\} \subseteq \Delta\sigma_t$, so $\Gamma\sigma_t \to \Delta\sigma_t \mathrel{\hat{\in}} N_t$ holds. Furthermore, if $\Gamma \to \Delta$ satisfies condition (ii) of Definition 4, then clause $\Gamma' \to \Delta' \in \mathcal{S}_v$ exists such that $\Gamma' \subseteq \Gamma$ and $\Delta' \subseteq \Delta$; but then, due to $\Gamma'\sigma_t \subseteq \Gamma\sigma_t \subseteq \Gamma_t$, we have that $\Gamma'\sigma_t \to \Delta'\sigma_t \in N_t$ holds, and so $\Gamma\sigma_t \to \Delta\sigma_t \mathrel{\hat{\in}} N_t$ holds as well. □

Lemma 6. *For each clause $\Gamma' \to \Delta' \vee s' \not\approx s' \in N_t$, we have $\Gamma' \to \Delta' \mathrel{\hat{\in}} N_t$.*

Proof. Consider an arbitrary clause $\Gamma' \to \Delta' \vee s' \not\approx s' \in N_t$. By the definition of N_t, a clause $\Gamma \to \Delta \vee s \not\approx s \in \mathcal{S}_t$ exists such that

$$\Gamma\sigma_t = \Gamma' \subseteq \Gamma_t, \quad \Delta\sigma_t = \Delta', \quad \text{and} \quad s\sigma_t = s'. \tag{6.24}$$

By the assumption of Theorem 2, the Ineq rule is not applicable to $\Gamma \to \Delta \vee s \not\approx s$, and so we have $\Gamma \to \Delta \mathrel{\hat{\in}} \mathcal{S}_v$. Since $\Gamma\sigma_t \subseteq \Gamma_t$, by Lemma 5 we have $\Gamma' \to \Delta' \mathrel{\hat{\in}} N_t$, as required. □

Lemma 7. *Both t and t' (if it exists) are irreducible by R_t.*

Proof. Recall that throughout Section 6.3.1, both t and t' are treated as constants, and that each rule $l \Rightarrow r \in R_t$ satisfies $l >_t r$ due to condition R2. Now if $t = c$, then t' does not exist and t the is smallest a-term in $>_t$ and therefore it cannot occur on the left-hand side of a rule in R_t, and so the lemma holds; hence, in the rest of this proof we assume that $t \neq c$. Condition L2 then ensures $t \approx t' \in \Delta_t$. Moreover, condition (i) of Definition 3 and condition O1 imply $t >_t t'$, and $t' \not>_t s$ for each a-term s. Therefore, t' is irreducible; moreover, to show that t is irreducible, we next prove that $t \Rightarrow t' \notin R_t$.

For the sake of a contradiction, assume that there exists a clause $\Gamma \to \Delta \vee t \approx t' \in N_t$ that generates the rule $t \Rightarrow t'$ in R_t. By the definition of N_t, we have $\Gamma \subseteq \Gamma_t$. Since $t \approx t' >_t \Delta$, we have $\Delta \subseteq \{t' \not\approx t', t' \approx t'\}$. Since the clause is generative, condition R1 ensures $t' \approx t' \notin \Delta$, and so we have $\Delta \subseteq \{t' \not\approx t'\}$. If $\Delta \neq \emptyset$, then Lemma 6 implies $\Gamma \to t \approx t' \hat{\in} N_t$, and otherwise $\Gamma \to t \approx t' \hat{\in} N_t$ holds by our assumption. But then, together with $t \approx t' \in \Delta_t$, this implies $\Gamma_t \to \Delta_t \hat{\in} N_t$, which contradicts condition L1. □

The proof of the following lemma is adapted from that of Nieuwenhuis and Rubio [NR95, Lemma 5.6].

Lemma 8. *For each $\Gamma \to \Delta \in N_t$, we have $R_t^* \models \Delta$.*

Proof. For the sake of a contradiction, choose $C^i = \Gamma^i \to \Delta^i \vee L^i$ as the clause in the sequence of clauses from Section 6.3.1.2 with the smallest i such that $R_t^* \not\models \Delta^i \vee L^i$; recall that $L^i >_t \Delta^i$ and $L^i = l^i \bowtie r^i$ with $\bowtie \in \{\approx, \not\approx\}$. Due to our choice of i, condition (i) of Lemma 4 holds for C^i and i. By the definition of N_t, a clause $\Gamma \to \Delta \vee L \in \mathcal{S}_v$ exists such that

$$\Gamma\sigma_t = \Gamma^i \subseteq \Gamma_t, \quad \Delta\sigma_t = \Delta^i, \quad L\sigma_t = L^i, \quad \text{and} \quad \Delta \not\succeq_v L. \tag{6.25}$$

We next prove the claim of this lemma by considering the possible forms of L^i.

Assume $L^i = l^i \approx r^i$ with $l^i = r^i$. But then, we have $R_t^* \models L^i$, which contradicts our assumption that $R_t^* \not\models \Delta^i \vee L^i$.

Assume $L^i = l^i \approx r^i$ with $l^i >_t r^i$. But then, literal L is of the form $l \approx r$ and it satisfies $l\sigma_t \approx r\sigma_t = l^i \approx r^i$. By the definition of $>_t$, we have $l \succ_v r$. We first show that $(R_t^{i-1})^* \not\models \Delta^i \vee L^i$ holds. Towards this goal, note that, for each equality $s_1 \approx s_2 \in \Delta^i \vee L^i$, properties $R_t^* \not\models s_1 \approx s_2$ and $R_t^{i-1} \subseteq R_t$ imply $(R_t^{i-1})^* \not\models s_1 \approx s_2$. Furthermore, for each inequality $s_1 \not\approx s_2 \in \Delta^i$, Lemma 2 and $R_t^* \not\models s_1 \not\approx s_2$ imply $(R_t^{i-1})^* \not\models s_1 \not\approx s_2$. Thus, clause C^i satisfies conditions R1 and R2; however, since $R_t^* \not\models l^i \approx r^i$, clause C^i is not generative and thus either condition R3 or condition R4 is not satisfied. We next consider both possibilities.

- Condition R3 does not hold—that is, l^i is reducible by R_t^{i-1}. By the definition of reducibility, a position p and a clause $C^j = \Gamma^j \to \Delta^j \vee l^j \approx r^j$ generating the rule $l^j \Rightarrow r^j$ exist such that $j < i$ and $l^i|_p = l^j$. By Lemma 7, l^j is neither t nor t', and therefore $l|_p$ is neither x

nor y. Due to $j < i$, we have $l^i \approx r^i \geq_t l^j \approx r^j$; together with $l^j \approx r^j >_t \Delta^j$, we have that $l^i \approx r^i >_t \Delta^j$. Lemma 3 ensures $R_t^* \not\models \Delta^j$, and the definition of N_t ensures that a clause $\Gamma' \to \Delta' \vee l' \approx r' \in \mathcal{S}_v$ exists such that

$$\Gamma'\sigma_t = \Gamma^j \subseteq \Gamma_t, \quad \Delta'\sigma_t = \Delta^j, \quad l'\sigma_t = l^j, \quad r'\sigma_t = r^j, \quad \Delta' \not\succeq_v l' \approx r', \quad \text{and} \quad l' \succ_v r'. \tag{6.26}$$

By the assumption of Theorem 2, the Eq rule is not applicable to (6.25) and (6.26), and so $\Gamma \wedge \Gamma' \to \Delta \vee \Delta' \vee l[r']_p \approx r \hat{\in} \mathcal{S}_v$. Let $\Delta'' = \Delta^i \vee \Delta^j \vee l^i[r^j]_p \approx r^i$. Then $\Gamma\sigma_t \cup \Gamma'\sigma_t \subseteq \Gamma_t$, so Lemma 5 ensures that $\Gamma^i \wedge \Gamma^j \to \Delta'' \hat{\in} N_t$ holds. Set R_t^* is a congruence, so $l^i[r^j]_p \approx r^i \notin R_t^*$ holds, and therefore $R_t^* \not\models \Delta''$ holds. Finally, $>_t$ is a simplification order, which ensures $l^i \approx r^i >_t l^i[r^j]_p \approx r^i$; together with $l^i \approx r^i >_t \Delta^i$ and $l^i \approx r^i >_t \Delta^j$, we have $l^i \approx r^i >_t \Delta''$. But then, Lemma 4 implies $R_t^* \models \Delta''$, which is a contradiction.

- Condition R4 does not hold. Then, there exists some term s such that $l^i \approx s \in \Delta^i$ and $s \approx r^i \in (R_t^{i-1})^*$. Due to $R_t^{i-1} \subseteq R_t$, we have $s \approx r^i \in R_t^*$, and so $R_t^* \not\models s \not\approx r^i$. Furthermore, $\Delta \vee L$ is of the form $\Delta' \vee l \approx r \vee l' \approx r'$ such that

$$l\sigma_t = l^i, \quad r\sigma_t = s, \quad l'\sigma_t = l^i, \quad \text{and} \quad r'\sigma_t = r^i. \tag{6.27}$$

We then have $l' = l$. By the assumption of Theorem 2, the Factor rule is not applicable to $\Gamma \to \Delta \vee L$, so $\Gamma \to \Delta' \vee r \not\approx r' \vee l' \approx r' \hat{\in} \mathcal{S}_v$. Let $\Delta'' = \Delta'\sigma_t \vee s \not\approx r^i \vee l^i \approx r^i$. But then, $\Gamma\sigma_t \subseteq \Gamma_t$ and Lemma 5 ensure that $\Gamma^i \to \Delta'' \hat{\in} N_t$ holds. By all these observations, we have $R_t^* \not\models \Delta''$. Moreover, $l^i >_t r^i$ and $l^i >_t s$ imply $l^i \approx r^i >_t s \approx r^i$; thus, $\Delta^i \vee l^i \approx r^i >_t \Delta''$ holds. Lemma 4 then implies $R_t^* \models \Delta''$, which is a contradiction.

Assume $L^i = l^i \not\approx r^i$ with $l^i = r^i$. Then, literal L is of the form $l \not\approx r$ such that it satisfies $l\sigma_t \not\approx r\sigma_t = l^i \not\approx r^i$. But then, $l^i = r^i$ implies $l = r$. By Lemma 6, we have $\Gamma^i \to \Delta^i \hat{\in} N_t$. Clearly, $\Delta^i \vee l^i \not\approx r^i >_t \Delta^i$, and so Lemma 4 implies $R_t^* \models \Delta^i$, which is a contradiction.

Assume $L^i = l^i \not\approx r^i$ with $l^i >_t r^i$. Lemma 2 ensures $(R_t^{i-1})^* \not\models l^i \not\approx r^i$ and hence l^i is reducible by R_t^{i-1}. Therefore, by the definition of reducibility, a position p and a generative clause $C^j = \Gamma^j \to \Delta^j \vee l^j \approx r^j$ exist such that both $j < i$ and $l^i|_p = l^j$. Since $j < i$, we have that $l^i \not\approx r^i >_t l^j \approx r^j >_t \Delta^j$. Lemma 3 ensures $R_t^* \not\models \Delta^j$, and the definition of N_t ensures that a clause $\Gamma' \to \Delta' \vee l' \approx r' \in \mathcal{S}_v$ exists satisfying (6.26), as in the first case. By the assumption of Theorem 2,

the Eq rule is not applicable to (6.25) and (6.26), and thus $\Gamma \wedge \Gamma' \to \Delta \vee \Delta' \vee l[r']_p \not\approx r \,\hat{\in}\, \mathcal{S}_v$ holds. Let $\Delta'' = \Delta^i \vee \Delta^j \vee l^i[r^j]_p \not\approx r^i$. We clearly have $\Gamma\sigma_t \cup \Gamma'\sigma_t \subseteq \Gamma_t$, and hence by Lemma 5 we have $\Gamma^i \wedge \Gamma^j \to \Delta'' \,\hat{\in}\, N_t$. Since R_t^* is a congruence, we have $R_t^* \not\models l^i[l^j]_p \not\approx r^i$, and therefore $R_t^* \not\models \Delta''$ holds. Finally, $>_t$ is a simplification order, so $l^i \not\approx r^i >_t l^i[l^j]_p$; together with $l^i \approx r^i >_t \Delta^i$ and $l^i \approx r^i >_t \Delta^j$, we have $l^i \approx r^i >_t \Delta''$. But then, Lemma 4 implies $R_t^* \models \Delta''$, which is a contradiction. □

Lemma 9. *For each clause $\Gamma \to \Delta$ with $\Gamma \to \Delta \,\hat{\in}\, N_t$, we have $R_t^* \models \Delta$.*

Proof. Apply Lemma 4 for $i = n + 1$ and Lemma 8. □

Lemma 10. *For each generative clause $\Gamma^i \to \Delta^i \vee l^i \approx r^i$, disjunction Δ^i does not contain a literal of the form $s \not\approx s$.*

Proof. For the sake of a contradiction, let us assume $C^i = \Gamma^i \to \Delta^i \vee l^i \approx r^i \in N_t$ is generative and that $s \not\approx s \in \Delta^i$ holds for some term s. By Lemma 6, we have $\Gamma^i \to \big(\Delta^i \setminus \{s \not\approx s\}\big) \vee l^i \approx r^i \,\hat{\in}\, N_t$, and so $\Gamma \to \Delta \in N_t$ holds for some $\Gamma \subseteq \Gamma^i$ and some $\Delta \subseteq \big(\Delta^i \setminus \{s \not\approx s\}\big) \cup \{l^i \approx r^i\}$. But then, Lemma 3 implies $R_t^* \not\models \Delta^i$; moreover, by condition R1, we have $(R_t^{i-1})^* \not\models \Delta^i \vee l^i \approx r^i$. We have the following two possibilities.

- $l^i \approx r^i \in \Delta$. Let j be the index of $\Gamma \to \Delta$ in the sequence of clauses from Section 6.3.1.2; clearly, $j < i$, and so $(R_t^{j-1})^* \subseteq (R_t^{i-1})^*$. Consider an arbitrary literal $L \in \Delta$: if L is of the form $l \approx r$, then $(R_t^{i-1})^* \not\models L$ clearly implies $(R_t^{j-1})^* \not\models L$; and if L is of the form $l \not\approx r$, then Lemma 2 applied for i ensures $R_t^* \not\models l \not\approx r$, and so Lemma 2 applied again for j ensures $(R_t^{j-1})^* \not\models l \not\approx r$. Thus, we have $(R_t^{j-1})^* \not\models \Delta$, and so $\Delta \subseteq \Delta^i$ clearly ensures that $\Gamma \to \Delta$ generates $l^i \Rightarrow r^i$. This, however, contradicts our assumption that C^i is generative.
- $l^i \approx r^i \notin \Delta$. But then, we have $\Delta \subseteq \Delta^i$, and so $R_t^* \not\models \Delta$. However, by Lemma 8 we have $R_t^* \models \Delta$, which is a contradiction. □

Lemma 11. $R_t^* \not\models \Gamma_t \to \Delta_t$.

Proof. We need to show both $R_t^* \models \Gamma_t$ and $R_t^* \not\models \Delta_t$.

($R_t^* \models \Gamma_t$) Note that condition L2 ensures $\Gamma_t \to A \,\hat{\in}\, N_t$, and so Lemma 9 ensures $R_t^* \models A$ for each atom $A \in \Gamma_t$.

($R_t^* \not\models \Delta_t$) Assume for the sake of a contradiction that a literal $L \in \Delta_t$ exists such that $R_t^* \models L$. Then, a generative clause $C^i = \Gamma^i \to \Delta^i \vee l^i \approx r^i \in N_t$ and a position p exist such that $L|_p = l^i$; let $\Delta = \Delta^i \vee l^i \approx r^i$. Since $>_t$ is a simplification order and $l^i >_t r^i$, we have $L \geq_t l^i \approx r^i$; but then, since $l^i \approx r^i >_t \Delta^i$, we have $L \geq_t \Delta$. We next consider an arbitrary literal $l \bowtie r \in \Delta$ with $\bowtie \in \{\approx, \not\approx\}$ and $l \geq_t r$; by the observations made thus far, $L \geq_t l \bowtie r$ holds. By condition O2, one of the following holds.

(i) $l \bowtie r \in \{t \approx t, t' \approx t'\}$. But then C^i is not generative by condition R1.

(ii) $l \bowtie r \in \{t \not\approx t, t' \not\approx t'\}$. But then C^i is not generative by Lemma 10.

(iii) $l \bowtie r = t \approx t'$. But then $t \approx t' \in \Delta_t$ by condition L2.

(iv) $l \bowtie r = t \not\approx t'$. By Lemma 7 both t and t' are irreducible by R_t, so $R_t^* \models t \not\approx t'$; by Lemma 2, then $(R_t^{i-1})^* \models t \not\approx t'$; but then C^i is not generative by condition R1.

(v) $l \approx r \in \Delta_t$ where $r = \mathsf{true}$.

Thus, (iii) and (v) are the only possible cases, and they both satisfy $l \bowtie r \in \Delta_t$. Since $l \bowtie r$ was arbitrarily chosen from Δ_t, we have $\Delta \subseteq \Delta_t$. But then, $\Gamma^i \subseteq \Gamma_t$ implies that $\Gamma_t \to \Delta_t \mathrel{\hat{\in}} N_t$ holds, which contradicts condition L1. □

6.3.2 Interpreting the Ontology $\mathcal{O}$

We now combine the rewrite systems R_t constructed in Section 6.3.1 into a single rewrite system R, and we then show that R^* satisfies $R^* \models \mathcal{O}$ and $R^* \not\models \Gamma_Q \to \Delta_Q$.

6.3.2.1 Unfolding the Context Structure

We construct R by a partial induction over the terms in HU_{a}. We define several partial functions: function X maps a term t to a context $X_t \in \mathcal{V}$; functions Γ and Δ assign to a term t a conjunction of atoms Γ_t and a disjunction of literals Δ_t, respectively; and function R maps each term into a model fragment R_t for t, X_t, Γ_t, and Δ_t.

M1. For the base case, we consider the constant c.

$$X_c = q \tag{6.28}$$

$$\Gamma_c = \Gamma_Q \sigma_c \tag{6.29}$$

$$\Delta_c = \Delta_Q \sigma_c \tag{6.30}$$

$$R_c = \text{the model fragment for } c, q, \Gamma_c, \text{ and } \Delta_c \tag{6.31}$$

M2. For the inductive step, assume that $X_{t'}$ has already been defined, and consider an arbitrary function symbol $f \in \Sigma_{\mathsf{a}}^F$ such that $f(t')$ is irreducible by $R_{t'}$. Let $u = X_{t'}$ and $t = f(t')$. We have two possibilities.

M2.a. Term t occurs in $R_{t'}$. Then, term $t = f(t')$ was generated in $R_{t'}$ by some ground clause $C = \Gamma \to \Delta \vee L \in N_{t'}$ such that $L >_t \Delta$ and $f(t')$ occurs in L. By the definition of N_t, then a clause $C' = \Gamma' \to \Delta' \vee L' \in \mathcal{S}_u$ exists such that $C = C'\sigma_{t'}$ and L' contains $f(x)$; moreover, $L >_{t'} \Delta$ implies $\Delta' \not\geq_u L'$. The Succ and Core rules are not applicable to $\mathcal{D}$, so we can choose a context $v \in \mathcal{V}$ such that $\langle u, v, f\rangle \in \mathcal{E}$ and $A \to A \,\hat{\in}\, \mathcal{S}_v$ for each $A \in K_2$, where K_2 is as in the Succ rule. We define the following:

$$X_t = v \tag{6.32}$$

$$\Gamma_t = R^*_{t'} \cap \mathsf{Su}_t \tag{6.33}$$

$$\Delta_t = \mathsf{Pr}_t \setminus R^*_{t'} \tag{6.34}$$

$$R_t = \text{the model fragment for } t, v, \Gamma_t, \text{ and } \Delta_t \tag{6.35}$$

M2.b. Term t does not occur in $R_{t'}$. Then, let $R_t = \{t \Rightarrow c\}$, and we do not define any other functions for t.

Finally, let R be the rewrite system defined by $R = \bigcup_t R_t$ where the union ranges over all R_t that have been defined above.

Lemma 12. *The model fragments R_c and R_t constructed in lines (6.31) and (6.35) satisfy conditions L1 through L3 in Section 6.3.1.*

Proof. The proof is by induction on the structure of terms $t \in \mathsf{dom}(X)$. For $t = c$, conditions L1

through L3 hold directly from conditions C1 through C3 of Theorem 2. We next assume that the lemma holds for some term $t' \in \mathsf{dom}(X)$, and we consider an arbitrary term t of the form $t = f(t')$; let $u = X_{t'}$ and $v = X_t$. Before proceeding, note that terms t and t' are irreducible by $R_{t'}$ due to condition M2; but then, since $\Gamma_t \subseteq R^*_{t'}$ holds by (6.33), each atom $A_i \in R_{t'}$ is generated by clause satisfying (6.36) (where subscript i does not necessarily indicate the position of the clause in sequence of clauses from Section 6.3.1.2). By the definition of $N_{t'}$, there exists a clause satisfying (6.37).

$$\Gamma_i \to \Delta_i \vee A_i \in N_{t'} \qquad \text{with} \quad A_i >_t \Delta_i \tag{6.36}$$

$$\Gamma'_i \to \Delta'_i \vee A'_i \in \mathcal{S}_u \qquad \Gamma_i = \Gamma'_i\sigma_{t'}, \quad \Delta_i = \Delta'_i\sigma_{t'}, \quad A_i = A'_i\sigma_{t'}, \quad \text{and} \quad \Delta'_i \not>_u A'_i \tag{6.37}$$

To prove condition L1, assume for the sake of a contradiction that $\Gamma_t \to \Delta_t \mathrel{\hat{\in}} N_t$ holds. We have $\Delta_t \subseteq \mathsf{Pr}_t$ due to (6.34). By condition (ii) of Definition 4, set N_t then contains a clause

$$\begin{array}{ll} \displaystyle\bigwedge_{i=1}^{m} A_i \to \bigvee_{i=m+1}^{m+n} L_i & \text{with} \quad \{\, A_i \mid 1 \leq i \leq m \,\} \subseteq \Gamma_t \\ & \text{and} \quad \{\, L_i \mid m+1 \leq i \leq m+n \,\} \subseteq \Delta_t \subseteq \mathsf{Pr}_t. \end{array} \tag{6.38}$$

By the definition of N_t, set $\mathcal{S}_v$ contains a clause

$$\begin{array}{ll} \displaystyle\bigwedge_{i=1}^{m} A'_i \to \bigvee_{i=m+1}^{m+n} L'_i & \text{with} \quad A_i = A'_i\sigma_t \text{ for } 1 \leq i \leq m+n \\ & \text{and} \quad L_i = L'_i\sigma_t \text{ and } L'_i \in \mathsf{Pr}(\mathcal{O}) \text{ for } m+1 \leq i \leq m+n. \end{array} \tag{6.39}$$

Each A_i with $1 \leq i \leq m$ is generated by a ground clause (6.36), and the latter is obtained from the corresponding non-ground clause (6.37). The Pred rule is not applicable to (6.37) and (6.39) so (6.40) holds; together with Lemma 5, this ensures (6.41).

$$\bigwedge_{i=1}^{m} \Gamma'_i \to \bigvee_{i=1}^{m} \Delta'_i \vee \bigvee_{i=m+1}^{m+n} A'_i\sigma \mathrel{\hat{\in}} \mathcal{S}_u \quad \text{for } \sigma = \{x \mapsto f(x),\ y \mapsto x\} \tag{6.40}$$

$$\bigwedge_{i=1}^{m} \Gamma_i \to \bigvee_{i=1}^{m} \Delta_i \vee \bigvee_{i=m+1}^{m+n} A_i \mathrel{\hat{\in}} N_{t'} \tag{6.41}$$

By Lemma 3, we have $R^*_{t'} \not\models \Delta_i$ for each $1 \leq i \leq m$; moreover, (6.34) ensures $R^*_{t'} \not\models \Delta_t$, which implies $R^*_{t'} \not\models L_i$ for each $m+1 \leq i \leq m+n$; however, this contradicts (6.41) and Lemma 9.

Next we show that condition L2 holds. Due to (6.34), we have $\Delta_t = \mathsf{Pr}_t \setminus R^*_{t'}$, and hence

$\Delta_t \subseteq \mathsf{Pr}_t$. Furthermore, $\{t \approx t'\} \subseteq \mathsf{Pr}_t$ holds by (6.21); moreover, t and t' are irreducible by $R_{t'}$, and so $t \approx t' \notin R^*_{t'}$ holds; consequently, we have $\{t \approx t'\} \subseteq \Delta_t$, as required.

Finally, we show that condition L3 holds. Consider an arbitrary atom $A_i \in \Gamma_t$, let (6.36) be the clause that generates A_i in $R_{t'}$, and let (6.37) be the corresponding non-ground clause. Since $A_i \in \mathsf{Su}_t$, atom A'_i is of the form $A''_i\sigma$, where σ is the substitution from the Succ rule; but then, $A''_i \in K_2$, where K_2 is as specified in the Succ rule. In condition M2.a we chose v so that the Succ rule is satisfied, and therefore $A''_i \to A''_i \hat{\in} \mathcal{S}_v$; but then, since $A''_i\sigma_t = A_i$, we have $A_i \to A_i \hat{\in} N_t$, as required for condition L3. □

6.3.2.2 Termination, Confluence, and Compatibility

Lemma 13. *The rewrite system R is Church-Rosser.*

Proof. We show that R is terminating and left-reduced, and thus Church-Rosser. In the proof of the former, we use a total simplification order $\rhd$ on all ground a- and p-terms defined as follows. We extend the precedence $\succcurlyeq$ from Definition 3 to all a- and p-symbols in an arbitrary way, but ensuring that constant true is smallest in the order; then, let $\rhd$ be a *lexicographic path order* [BN98] over such $\succcurlyeq$. It is well known that such $\rhd$ is a simplification order, and that it satisfies the following properties for each a-term t with predecessor t' (if it exists), all function symbols $f, g \in \Sigma^F_{\mathsf{a}}$, and each p-term A:

- $f(t) \rhd t \rhd t'$,
- $f \succcurlyeq g$ implies $f(t) \rhd g(t)$, and
- $A \rhd \mathsf{true}$.

Thus, conditions (i) and (ii) of Definition 3 and the manner in which context orders are grounded in Section 6.3.1.1 clearly ensure that, for each a-term $t \in \mathsf{dom}(X)$ and for all terms s_1 and s_2 from the a-neighbourhood of t with $s_1 >_t s_2$, we have $s_1 \rhd s_2$.

We next show that R is terminating by arguing that each rule in R is embedded in $\rhd$. To this end, consider an arbitrary rule $l \Rightarrow r \in R$. Clearly, a term $t \in \mathsf{dom}(R)$ exists such that $l \Rightarrow r \in R_t$. This rule is obtained from a head literal $l \approx r$ of a clause in N_t, and condition R2 of the definition of R_t ensures that $l >_t r$. Moreover, $l \approx r$ is obtained by grounding a context literal with σ_t, so we have the following possible forms of $l \approx r$.

- Terms l and r are both from the a-neighbourhood of t. Then, $l >_t r$ implies $l \rhd r$.
- We have $l \approx r = A \approx \mathsf{true}$ for A a p-term. Then, $A \rhd \mathsf{true}$ since true is smallest in $\succcurlyeq$.

We next show that R is left-reduced. For the sake of a contradiction, assume that a rule $l \Rightarrow r \in R$ exists such that l is reducible by $R' = R \setminus \{l \Rightarrow r\}$. Let p be the 'deepest' position at which some rule in R' reduces l (i.e., no rule in R' reduces l at position below p), and let $l' \Rightarrow r' \in R'$ be the rule that reduces l at position p; thus, $l' = l|_p$. By the definition of R, term t exists such that $l' \Rightarrow r' \in R_t$ where t can be as follows.

- Term t is handled in condition M2.a. Then $l' \Rightarrow r'$ is generated by an equality $l' \approx r'$ in the head of a generative context clause, and so $l' \approx r'$ is of the form $f(t) \approx g(t)$, $f(t) \approx t$, $f(t) \approx t'$, or $t \approx t'$. Thus, $f(t)$ is reducible by R_t, which contradicts condition M2 from the construction of R.
- Term t is handled in condition M2.b. Then $l' = t$; moreover, R' does not contain t by the construction of R, which contradicts the assumption that $l' \Rightarrow r' \in R'$. □

Lemma 14. *For each term t, each $f \in \Sigma^F_{\mathsf{a}}$, and each atom $A \in \mathsf{Su}_t \cup \mathsf{Pr}_{f(t)} \cup \mathsf{Ref}_t$ such that $A \in R^*$ and all* a*-terms in A are irreducible by R, we have $A \in R^*_t$.*

Proof. Let t be a term, let $f \in \Sigma^F_{\mathsf{a}}$ be a function symbol, and let $A \in \mathsf{Su}_t \cup \mathsf{Pr}_{f(t)} \cup \mathsf{Ref}_t$ be an atom such that all a-terms in A are irreducible by R; the latter and $A \in R^*$ ensure that $A \Rightarrow \mathsf{true} \in R$. We next consider the possible forms of A.

Assume $A \in \mathsf{Su}_t$. By the definition of Su_t in (6.20) and the fact that $\mathsf{Su}(\mathcal{O})$ contains only atoms of the form $B(x)$, $S(x,y)$, and $S(y,x)$, atom A can be of the form $B(t)$, $S(t,t')$, or $S(t',t)$, for t' the predecessor of t (if it exists). Due to the form of context literals in Definition 1 and the definition of grounding from Section 6.3.1, atom A can occur only in N_t or $N_{t'}$; therefore, we have $A \in R^*_t$ or $A \in R^*_{t'}$. Now assume $A \in R^*_{t'}$. Due to $A \in \mathsf{Su}_t$ and the definition of Γ_t in (6.33), we have $A \in \Gamma_t$. Lemma 11 ensures that $R^* \not\models \Gamma_t \to \Delta_t$. But then, we have $A \in R^*_t$, as required.

Assume $A \in \mathsf{Pr}_{f(t)}$. By the definition of $\mathsf{Pr}_{f(t)}$ in (6.21) and the fact that $\mathsf{Pr}(\mathcal{O})$ contains only atoms of the form $B(y)$, $S(y,x)$, and $S(x,y)$, atom A can be of the form, $B(t)$, $S(t,f(t))$, or $S(f(t),t)$; note that $x \approx y \in \mathsf{Pr}(\mathcal{O})$ is not an atom and is therefore not relevant to this analysis. Due to the form of context literals in Definition 1 and the definition of grounding from Section 6.3.1, atom A can occur only in N_t or $N_{f(t)}$; therefore, $A \in R^*_t$ or $A \in R^*_{f(t)}$. Assume for

the sake of a contradiction that $A \notin R^*_t$, but $A \in R^*_{f(t)}$. Due to $A \in \mathsf{Pr}_{f(t)}$ and the definition of $\Delta_{f(t)}$ in (6.34), we have $A \in \Delta_{f(t)}$; due to Lemma 11, we have $R^*_{f(t)} \not\models \Gamma_{f(t)} \rightarrow \Delta_{f(t)}$; therefore, we have $A \notin R^*_{f(t)}$, which is a contradiction.

Assume $A \in \mathsf{Ref}_t$. Due to the form of context literals in Definition 1 and the definition of grounding from Section 6.3.1, atom A can occur only in N_t, and so $A \in R^*_t$. □

Lemma 15. *Let s_1 and s_2 be both DL-*a*-terms or both DL-*p*-terms, and let τ be a ground substitution irreducible by R such that $\tau(x)$ is defined, $s_1\tau$ and $s_2\tau$ are ground, and each $\tau(z_i)$ (if defined) is in the* a*-neighbourhood of $\tau(x)$. Then, for $\bowtie \in \{\approx, \not\approx\}$, if $R^*_{\tau(x)} \models s_1\tau \bowtie s_2\tau$, then $R^* \models s_1\tau \bowtie s_2\tau$.*

Proof. Let s_1 and s_2 and τ be as stated above, let $t = \tau(x)$, and let t' be the predecessor of t (if it exists). Since t is irreducible by R, rewrite system R_t has been defined in Section 6.3.2.1. We next consider the possible forms of $\bowtie$.

- Assume $\bowtie = \approx$. But then, $R_t \subseteq R$ and $R^*_t \models s_1\tau \approx s_2\tau$ imply $R^* \models s_1\tau \approx s_2\tau$.
- Assume $\bowtie = \not\approx$. Let s'_1 and s'_2 be the normal forms of $s_1\tau$ and $s_2\tau$, respectively, w.r.t. R_t. Due to the shape of DL-literals, s_1 and s_2 can be of the form $f(x)$, x, or z_i; therefore, terms $s_1\tau$ and $s_2\tau$ are of the form $f(t)$, t, or t', and terms s'_1 and s'_2 are of the form $g(t)$, t, or t'. Term t is irreducible by R, and thus t' is irreducible by R as well. Furthermore, $g(t)$ is irreducible by R_t and R_t is the only rewrite system where $g(t)$ can occur on the left-hand side of a rewrite rule, so therefore $g(t)$ is irreducible by R as well. But then, s'_1 and s'_2 are the normal forms of $s_1\tau$ and $s_2\tau$, respectively, w.r.t. R; thus, $R^* \models s'_1 \not\approx s'_2$, and therefore $R^* \models s_1\tau \not\approx s_2\tau$ holds, as required. □

6.3.2.3 The Completeness Claim

Lemma 16. *For each DL-clause $\Gamma \rightarrow \Delta \in \mathcal{O}$, we have $R^* \models \Gamma \rightarrow \Delta$.*

Proof. Consider an arbitrary DL-clause $\Gamma \rightarrow \Delta \in \mathcal{O}$ of the following form:

$$\bigwedge_{i=1}^{n} A_i \rightarrow \Delta \tag{6.42}$$

Let τ' be an arbitrary substitution such that $\Gamma\tau' \rightarrow \Delta\tau'$ is ground, and let τ be the substitution obtained from τ' by replacing each ground term with its normal form w.r.t. R. Since R^* is a con-

gruence, we have $R^* \models \Gamma\tau' \to \Delta\tau'$ if and only if $R^* \models \Gamma\tau \to \Delta\tau$. We next assume that $R^* \models \Gamma\tau$, and we show that $R^* \models \Delta\tau$ holds as well.

Consider an arbitrary atom $A_i \in \Gamma$. By the definition of DL-clauses, A_i is of the form $B(x)$, $S(x,x)$, $S(x,z_j)$, or $S(z_j,x)$. Substitution τ is irreducible by R, and so all a-terms in $A_i\tau$ are irreducible by R; but then, $A_i\tau \in R^*$ clearly implies $A_i\tau \Rightarrow \mathsf{true} \in R$. Each such rule is obtained from a generative clause so $A_i\tau$ is of the form $B(t)$, $S(t,t)$, $S(t,f(t))$, $S(f(t),t)$, $S(t,t')$, or $S(t',t)$, where $t = \tau(x)$ and t' is the predecessor of t (if it exists). We next prove that $A_i\tau \in \mathsf{Su}_t \cup \mathsf{Pr}_{f(t)} \cup \mathsf{Ref}_t$ holds by considering the possible forms of A_i.

- $A_i = B(x)$, so $A_i\tau = B(t)$. Then, $B(x) \in \mathsf{Su}(\mathcal{O})$, and so $B(t) \in \mathsf{Su}_t$ holds.
- $A_i = S(x,x)$, so $A_i\tau = S(t,t)$. Then, we clearly have $S(t,t) \in \mathsf{Ref}_t$.
- $A_i = S(x,z_j)$, so $A_i\tau$ is of the form $S(t,t')$ or $S(t,f(t))$. Then, $S(x,y) \in \mathsf{Su}(\mathcal{O})$, and so $S(t,t') \in \mathsf{Su}_t$ holds; moreover, $S(y,x) \in \mathsf{Pr}(\mathcal{O})$, and so $S(t,f(t)) \in \mathsf{Pr}_{f(t)}$ holds.
- $A_i = S(z_j,x)$, so $A_i\tau$ is of the form $S(t',t)$ or $S(f(t),t)$. Then, $S(y,x) \in \mathsf{Su}(\mathcal{O})$, and so $S(t',t) \in \mathsf{Su}_t$ holds; moreover, $S(x,y) \in \mathsf{Pr}(\mathcal{O})$, and so $S(f(t),t) \in \mathsf{Pr}_{f(t)}$ holds.

Lemma 14 then implies $A_i\tau \in R_t$, and so N_t contains a generative clause of the form (6.43). Now let $v = X_t$; by the definition of N_t, set $\mathcal{S}_v$ contains a clause of the form (6.44).

$$\Gamma_i \to \Delta_i \vee A_i \text{ with } A_i >_t \Delta_i \text{ and } \Gamma_i \subseteq \Gamma_t \tag{6.43}$$

$$\Gamma'_i \to \Delta'_i \vee A'_i \text{ with } \Delta'_i \not\geq_v A'_i \text{ and } \Gamma'_i\sigma_t = \Gamma_i,\ \Delta'_i\sigma_t = \Delta_i, \text{ and } A'_i\sigma_t = A_i \tag{6.44}$$

The Hyper rule is not applicable to (6.42) and (6.44), and therefore (6.45) holds, where σ is the substitution obtained from τ by replacing each occurrence of t (possibly nested in another term) with x. Finally, Lemma 5 ensures that (6.46) holds as well.

$$\bigwedge_{i=1}^{n} \Gamma'_i \to \Delta\sigma \vee \bigvee_{i=1}^{n} \Delta'_i \mathrel{\hat{\in}} \mathcal{S}_v \tag{6.45}$$

$$\bigwedge_{i=1}^{n} \Gamma_i \to \Delta\tau \vee \bigvee_{i=1}^{n} \Delta_i \mathrel{\hat{\in}} N_t \tag{6.46}$$

Now (6.46) and Lemma 9 imply $R^*_t \models \Delta\tau \vee \bigvee_{i=1}^{n} \Delta_i$, but Lemma 3 implies $R^*_t \not\models \Delta_i$; therefore, we have $R^*_t \models \Delta\tau$. Finally, Lemma 15 ensures $R^* \models \Delta\tau$, as required. □

Lemma 17. $R^* \not\models \Gamma_Q \rightarrow \Delta_Q$.

Proof. The claim is equivalent to proving $R^* \not\models \Gamma_c \rightarrow \Delta_c$. Lemma 11 implies $R^*_c \not\models \Gamma_c \rightarrow \Delta_c$, and so $R^*_c \models \Gamma_c$ and $R^*_c \not\models \Delta_c$ hold. The former observation and Lemma 15 ensure that $R^* \models \Gamma_c$ holds. Furthermore, for each atom $B(x) \in \Delta_Q$, Definition 2 ensures $B(y) \in \mathsf{Pr}(\mathcal{O})$, and so $B(c) \in \mathsf{Pr}_{f(c)}$ holds for each $f \in \Sigma^F_{\mathsf{a}}$; hence, the contrapositive of Lemma 14 ensures $R^* \not\models B(c)$. Finally, for each atom $S(x,x) \in \Delta_Q$, we have $S(c,c) \in \mathsf{Ref}_c$; hence, the contrapositive of Lemma 14 ensures $R^* \not\models S(c,c)$. Consequently, $R^* \not\models \Delta_c$ holds, as required. □

6.4 Proofs of Worst-Case Optimality and Pay-As-You-Go Behaviour

Proposition 2. *For each expansion strategy that introduces at most exponentially many contexts, algorithm A1–A4 runs in worst-case exponential time.*

Proof. The number $\wp$ of context clauses that can be generated using the symbols in $\mathcal{O}$ is at most exponential in the size of $\mathcal{O}$, and the number m of clauses participating in each inference is linear in the size of $\mathcal{O}$. Hence, with k contexts, the number of inferences is bounded by $(k \cdot \wp)^m$; if k is at most exponential in the size of $\mathcal{O}$, the number of inferences is exponential as well. Thus, if at most exponentially many contexts are introduced, our algorithm runs in exponential time. □

Proposition 3. *For $\mathcal{ELH}$ ontologies and queries of the form $B_1(x) \rightarrow B_2(x)$, algorithm A1–A4 runs in polynomial time with either the cautious or the eager strategy; and with the cautious strategy and the* Hyper *rule applied eagerly, the inferences in step A3 correspond directly to the inferences of the $\mathcal{ELH}$ calculus by Baader et al. [BBL05].*

Proof. Consider an $\mathcal{ELH}$ ontology that is transformed into a set $\mathcal{O}$ of DL-clauses as specified in Chapter 2, and consider a query of the form $B_1(x) \rightarrow B_2(x)$. Due to the form of the query, the core of q is initialised to $B_1(x)$.

We first consider applying algorithm A1–A4 to $\mathcal{O}$ with the cautious strategy and the eager application of the Hyper rule. By induction on the application of the inference rules from Table 5.1, we next show that each context clause derived by the inference rules is of the form (6.47)

through (6.51) and that the core of each context is of the form $B(x)$.

$$\top \to B(x) \tag{6.47}$$

$$\top \to S(x, f(x)) \tag{6.48}$$

$$\top \to B(f(x)) \tag{6.49}$$

$$S(y, x) \to B(y) \tag{6.50}$$

$$S_1(y, x) \to S_2(y, x) \tag{6.51}$$

In particular, in step A3 we can perform the following inferences, with the specified correspondence to the completion rules **CR1**–**CR4** and **CR10** by Baader et al. [BBL05].

- The core of each context is of the form $B(x)$, so the Core rule introduces a clause of the form (6.47). This corresponds to way in which Baader et al. [BBL05] initialise their mappings.
- The Hyper rule can be applied to a DL-clause of type DL1. All other clauses participating in the inference are of the form (6.47), so the result is of the form (6.47). Such an inference corresponds to the completion rules **CR1** or **CR2**.
- The Hyper rule can be applied to a DL-clause of type DL2. The other clause participating in the inference is of the form (6.47), so the result is of the form (6.48) or (6.49). Moreover, function symbol f occurs in $\mathcal{O}$ in exactly one pair of clauses of type DL2, and the Hyper rule is applied eagerly; thus, whenever f occurs in a context in a clause of the form (6.48), it also occurs in a clause of the form (6.49). Now the Succ rule can be applied to the function symbol f, in which case the cautious strategy thus returns a context whose core is of the form $B(x)$. All of these inferences correspond to the completion rule **CR3**.
- The Hyper rule can be applied to a DL-clause of type DL3. The two other clauses participating in the inference are of the form (6.50) and (6.47), so the result is of the form (6.50); the Pred rule can then be applied to the latter clause, producing a clause of the form (6.47). Such a pair of inferences corresponds to the completion rule **CR4**.
- The Hyper rule can be applied to a DL-clause of type DL5. The other clause participating in the inference is of the form (6.50), so the result is of the form (6.50) as well; the Pred rule can then be applied to the latter clause, producing a clause of the form (6.48). Such a pair of inferences corresponds to the completion rule **CR10**.

One can show in an analogous way that each inference of the calculus by Baader et al. [BBL05] corresponds to one or more inferences of our calculus. Furthermore, it is clear that our algorithm runs in polynomial time.

We next consider applying algorithm A1–A4 to $\mathcal{O}$ with the eager strategy. One can show that the core of each context is of the form $A(x)$, $R(y,x)$, or $A(x) \wedge R(y,x)$, and that context can contain clauses of the form (6.47) through (6.53).

$$\top \to S_2(y,x) \tag{6.52}$$

$$\top \to B(y) \tag{6.53}$$

The proof is analogous to the case of the cautious strategy (without correspondence to the completion rules) so we omit the details for the sake of brevity. The only minor difference is that, if an application of the Pred rule to contexts u and v introduces a clause of the form (6.48) in u, then the Succ rule does not become applicable to u since the precondition of the Succ rule is still satisfied by v. Thus, the Succ rule never introduces contexts whose cores contain conjunctions of binary atoms. Thus, if $\mathcal{O}$ contains k_1 unary and k_2 binary predicates, the number of contexts is bounded by $O(k_1 \cdot k_2)$, and each context can contain at most $k_1 + k_2 + k_1 \cdot k_2$ clauses. All inference rules can be applied in polynomial time, so the algorithm runs in polynomial time. □

Part III

Practical Considerations

Chapter 7

System Description

To validate the practicality of our calculus, we implemented it in a new reasoning system called *Sequoia*. The system currently handles the $\mathcal{SRIQ}$ subset of OWL 2 DL (i.e., it does not support datatypes and nominals), for which it can decide concept satisfiability and concept subsumption and can classify an ontology. Other standard services such as ABox realisation are currently not supported. The system is written in Scala. In this chapter, we describe the architecture of the system and discuss certain aspects of the design that proved critical to the performance of the system.

7.1 System Architecture

The architecture of Sequoia is shown in Figure 7.1. It can be used via the command line, through the Protégé plug-in [KFNM04], or directly with the OWL API [HB11]. OWL API objects are translated to Sequoia's own representation of OWL axioms and expressions which does not depend on the OWL API. This has the advantage that the core reasoner implementation is independent from (potentially breaking) changes to the OWL API or its dependencies between versions. The *Reasoning Engine* is the core component of Sequoia, and consists of a sequence of pipeline phases; arrows show the flow of data through the pipeline.

The *chain encoding* phase encodes away complex role inclusion axioms in the input ontology using finite automata in the standard manner [HKS06]. The *clausification* phase is responsible for structurally transforming a $\mathcal{SRIQ}$ ontology into a set of DL-clauses (see Section 2.6 in Chapter 2), which are indexed by the subsequent *ontology indexing* phase (as described in Sec-

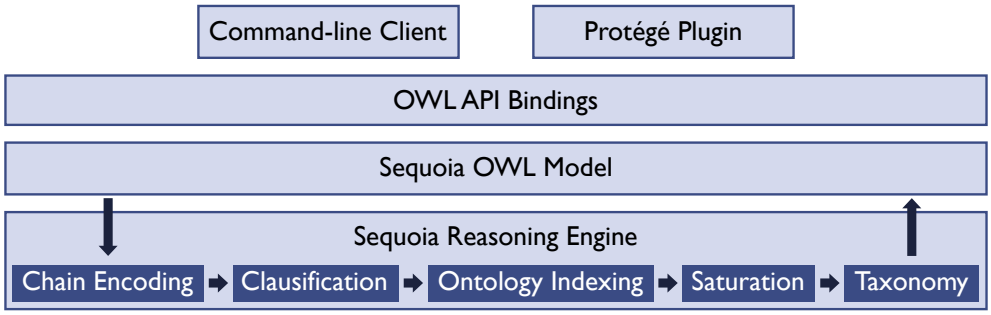


Figure 7.1: Main components of the Sequoia reasoner and data flow during classification

tion 7.3.2). The *saturation* phase implements the core reasoning algorithm from Chapter 5 using the techniques described in the rest of this chapter. The result of the saturation phase is a context structure $\mathcal{D}$ to which no inference rule is applicable. Finally, the *taxonomy* phase collects the subsumption information from the saturated context structure $\mathcal{D}$ and produces the (transitively reduced) concept hierarchy as output.

7.2 Saturation Implementation

In this section, we discuss how Sequoia saturates the context structure $\mathcal{D}$ under the inference rules of the calculus. Our saturation procedure is inspired by the algorithm implemented by Prover9 [McC10], the Otter theorem prover [McC94, MW97], and related early systems [RV03].

The pseudocode of our saturation algorithm is shown Algorithm 1. The procedure is parameterised by the input ontology $\mathcal{O}$, the context structure $\mathcal{D}$ to be saturated, and the set inference rules of the calculus $\mathcal{R}$ from Table 5.1 that derive new consequences other than the Ineq rule, which is handled as a special case (i.e., all inference rules other than the Ineq and Elim rules). We assume that the context structure $\mathcal{D}$ has been initialised as required by Theorem 2. The procedure maintains a set of triples called unprocessed. Each triple $\langle v, C, r \rangle$ represents that clause C in context v has not yet participated in the inference rule r.

The algorithm first initialises set unprocessed so that each inference rule must be applied to each clause in each context (line 1). Next, the algorithm enters a loop (lines 2 to 13) in which it iteratively processes unprocessed. In each iteration, a triple $\langle v, C, r \rangle$ is selected from unprocessed (line 3); we discuss the selection strategy in Section 7.2.1. The algorithm next removes the triple from unprocessed and adds clause C to $\mathcal{S}_v$ (lines 4 and 5), and then it computes the context u

Algorithm 1 The context structure saturation algorithm of Sequoia

Input:
$\mathcal{O}$: the input ontology
$\mathcal{D}$: the context structure $\mathcal{D} = \langle \mathcal{V}, \mathcal{E}, \mathcal{S}, \mathsf{core}, \succ \rangle$ to be saturated
$\mathcal{R}$: the set of inference rules of the calculus other than the Ineq and Elim rules
Output:
the context structure $\mathcal{D}$ saturated under $\mathcal{R}$ with respect to $\mathcal{O}$

```
1:  unprocessed ← { ⟨v, C, r⟩ | v ∈ V and C ∈ S_v and r ∈ R }
2:  while unprocessed ≠ ∅ do
3:      ⟨v, C, r⟩ ← SELECT(unprocessed)
4:      unprocessed ← unprocessed \ { ⟨v, C, r⟩ }
5:      S_v ← S_v ∪ {C}
6:      ⟨u, conclusions⟩ ← COMPUTECONCLUSIONS(O, D, v, C, r)
7:      for each D_1 ∈ conclusions do
8:          existing ← S_u ∪ { D_2 | ⟨u, D_2, w⟩ ∈ unprocessed for some w ∈ R }
9:          D_1 ← SIMPLIFYCLAUSE(D_1, existing)
10:         if D_1 ≠ null and there does not exist D_2 ∈ existing such that REDUNDANT(D_1, D_2) then
11:             S_u ← { D_2 ∈ S_u | not REDUNDANT(D_2, D_1) }
12:             unprocessed ← { ⟨u', D_2, w⟩ ∈ unprocessed | if u' = u then not REDUNDANT(D_2, D_1) }
13:             unprocessed ← unprocessed ∪ { ⟨u, D_1, w⟩ | w ∈ R }
14: return D
```

and the set conclusions of clauses obtained by applying inference rule r to clause C in context v (line 6); we discuss this step in Section 7.2.2. For each clause D_1 obtained in this way, the algorithm tries to simplify D_1 (line 9) and determine whether D_1 is forward redundant in the set of clauses derived thus far (line 10). If D_1 is not redundant, then the algorithm removes from $\mathcal{S}_u$ and unprocessed all clauses that are redundant due to D_1 (lines 11 and 12). We discuss the simplification and redundancy elimination steps in Section 7.2.3. Finally, the algorithm schedules all inference rules of $\mathcal{R}$ to be applied to D_1 in context u (line 13). The algorithm terminates when unprocessed becomes empty, at which point the context structure $\mathcal{D}$ is saturated under $\mathcal{R}$.

7.2.1 Selecting an Unprocessed Triple

We now describe our strategy for selecting the next triple $\langle v, C, r \rangle$ to process from set unprocessed (line 3) in each iteration.

By Proposition 3 from Chapter 5, the Hyper rule must be applied eagerly for our calculus to be worst-case optimal on $\mathcal{ELH}$ ontologies. Furthermore, our practical experience suggests that delaying the application of the Succ and Pred rules as long as possible can improve the performance. Finally, in order to (i) reduce the number of clauses retained during forward redundancy checking and (ii) increase the number of clauses removed during backward redundancy elimi-

nation, we select Horn clauses with empty bodies before Horn clauses with non-empty bodies, and we select Horn clauses before non-Horn clauses.

Thus, each inference rule r is assigned an integer priority $\rho(r)$, and each clause C is assigned an integer priority $\rho(C)$ according to the rules outlined in the previous paragraph. Function SELECT chooses a triple $\langle v, C, r\rangle$ such that, for all other $\langle v', C', r'\rangle$ in unprocessed, we have $\rho(r) \geq \rho(r')$ and if $\rho(r) = \rho(r')$ then $\rho(C) \geq \rho(C')$. For the fast retrieval of triple a satisfying these conditions, unprocessed is implemented as a multilevel queue.

7.2.2 Computing Conclusions of the Rule

Procedure COMPUTECONCLUSIONS$(\mathcal{O}, \mathcal{D}, v, C, r)$ applies the inference rule r to the clause C in the context v (possibly using other clauses from $\mathcal{S}_v$ and the ontology $\mathcal{O}$) in all possible ways. The result is a pair $\langle u, \mathsf{conclusions}\rangle$ where u is a context and conclusions is a set of clauses such that $C \hat{\in} \mathcal{S}_u$ must hold for each clause $C \in \mathsf{conclusions}$.

Note that if r is one of Core, Hyper, Eq, Ineq, or Factor, then $u = v$ holds by the definition of the inference rule r. Furthermore, if r is the Succ rule, then for some f there exists a possibly newly introduced edge $\langle v, u, f\rangle \in \mathcal{E}$. Finally, if r is the Pred rule, then for some f there exists either the edge $\langle u, v, f\rangle \in \mathcal{E}$ or $\langle v, u, f\rangle \in \mathcal{E}$.

7.2.3 Simplification, Forward Redundancy Checking and Backward Redundancy Elimination

Simplification rules are applied to each conclusion $D_1 \in \mathsf{conclusions}$ of the form $D_1 = \Gamma \to \Delta$ in an attempt to either remove unnecessary literals from Δ or to discard the clause completely. First, in line 8 the algorithm computes the set existing of both processed and unprocessed clauses in context u. Then, in line 9 the algorithm calls SIMPLIFYCLAUSE$(D_1, \mathsf{existing})$, which implements the simplification rules described below.

- The Ineq rule is applied to D_1 as a simplification rule rather than as a separate inference rule—if $t \not\approx t \in \Delta$ holds for some term t, then $t \not\approx t$ is immediately deleted from Δ. This reduces the total number of separate inference rule applications (since the result of the Ineq rule always makes the original clause redundant), and the resultant clause could remove more clauses during backward redundancy elimination.

- If $\{s \approx s',\ s \not\approx s'\} \subseteq \Delta$ or $s \approx s \in \Delta$ holds for some terms s and s', then D_1 is immediately discarded (i.e., the call to SIMPLIFYCLAUSE returns *null*) in accordance with condition (i) of Definition 4.
- A simplified form of *equational literal cutting* [Sch13b, Sch02] is applied: if $s \approx t \in \Delta$ holds for some terms s and t and $\top \rightarrow s \not\approx t \in$ existing, then $s \approx t$ is deleted from Δ; similarly, if $s \not\approx t \in \Delta$ holds for some terms s and t and $\top \rightarrow s \approx t \in$ existing, then $s \not\approx t$ is deleted from Δ. These optimisations are known in the literature as *negative simplify-reflect* and *positive simplify-reflect*, respectively, and they correspond to a combination of the Eq and Elim rules.

Next, in line 10 the algorithm applies the *forward redundancy check*. In particular, procedure REDUNDANT$(\Gamma_1 \rightarrow \Delta_1, \Gamma_2 \rightarrow \Delta_2)$ returns *true* if $\Gamma_2 \subseteq \Gamma_1$ and $\Delta_2 \subseteq \Delta_1$ (cf. condition (ii) of Definition 4); in such a case, we say that $\Gamma_2 \rightarrow \Delta_2$ is *stronger* than $\Gamma_1 \rightarrow \Delta_1$, and conversely $\Gamma_1 \rightarrow \Delta_1$ is *weaker* than $\Gamma_2 \rightarrow \Delta_2$. Then, forward redundancy involves checking whether existing contains a clause D_2 that is stronger than D_1. Finally, if the simplified clause is not forward redundant, the algorithm next applies *backward redundancy elimination* (lines 11 and 12) by deleting from $\mathcal{S}_u$ and unprocessed all clauses that are weaker than D_1 (and thus implements the Elim rule). Both forward and backward redundancy detection require searching potentially large sets of clauses so, to optimise this search, Sequoia maintains a redundancy index that we describe in Section 7.3.5.

7.2.4 Discussion

Practical experiments have shown that it is beneficial to spend time simplifying and removing redundant clauses whenever possible. This can considerably reduce the memory consumption (thus possibly avoiding memory exhaustion) and can lead to important simplifications being applied earlier (thus reducing the total number of generated clauses). Furthermore, our experience shows that, for ontologies encountered in practice, the contexts generated by our calculus are usually satisfiable. Hence, eager simplification and redundancy elimination are both important because our system, unlike general first-order theorem provers, is generating a saturated set of clauses (for each context) instead of searching for a refutation.

We could have used a saturation procedure where clauses cannot simplify or be simplified until they are selected on line 3. This form of delayed simplification is known as the Discount

loop [RV03] and it is used in the KAON2 reasoner [Mot06]. The tradeoff between the two techniques is that our approach spends much more time simplifying with the possible benefit of making an important simplification sooner [McC10].

7.2.5 Defining the Context Term Order

The conditions on the context term order $\succ_v$ for a context v as imposed by Definition 3 are required to ensure completeness. However, these conditions still provide some room for fine-tuning the actual order used in the implementation. For example, given atoms $B_1(t_1)$ and $B_2(t_2)$ such that $\{B_1(t_1), B_2(t_2)\} \cap \mathsf{Pr}(\mathcal{O}) = \emptyset$, we can define $B_1(t_1) \succ_v B_2(t_2)$ to hold according to either of the following rules:

(i) $B_1 \gtrdot B_2$, or $B_1 = B_2$ and $t_1 \succ_v t_2$; or

(ii) $t_1 \succ_v t_2$, or $t_1 = t_2$ and $B_1 \gtrdot B_2$.

Note that both of these orders are valid according to Definition 3.

If we choose rule (ii), then $A(f(x)) \succ_v B(x)$ holds regardless of the relative order of A and B in the $\gtrdot$ order, whereas this is not the case for rule (i). Practical experiments have shown that in many cases it is beneficial to use a context term order that adheres to rule (ii) instead of rule (i): each unary atom occurring in a DL-clause body must be of the form $C(x)$, and the substitution σ used in the Hyper rule has the property that $\sigma(x) = x$, and therefore an order that adheres to rule (ii) can reduce the number of applications of the Hyper rule in a context. This design principle can be applied to both unary and binary atoms.

An order that adheres to rule (ii) effectively puts symbols of the abstract sort a higher in the order $\gtrdot$ than symbols of the predicate sort p when constructing the lexicographic path order. Note that this is the opposite of the convention commonly used in implementations of superposition calculi, where symbols of the predicate sort are higher in the order [NR01, Section 3.2].

Note, however, that if $q \in \mathcal{V}$ is a context that is to be used for query clause entailment checking, then the context term order $\succ_q$ must additionally satisfy the conditions imposed by Theorem 2, and thus the ordering conditions must be appropriately relaxed.

Furthermore, practical experience has shown that it is also important that we choose the precedence of symbols of the predicate sort p in the order $\gtrdot$ to be such that auxiliary atoms introduced during structural transformation are smallest in the order. For example, suppose we

have an ontology $\mathcal{O}$ containing the following DL-clauses, where $T_1, \ldots, T_n$ are fresh predicates introduced during structural transformation:

$$\begin{aligned} \top &\rightarrow T_1(x) \vee \ldots \vee T_n(x) && \\ T_i(x) &\rightarrow A_i(x) && \text{for } 1 \leq i \leq n \\ T_i(x) &\rightarrow B_i(x) && \text{for } 1 \leq i \leq n \end{aligned}$$

Note that the fresh predicates occur in both heads and bodies of clauses. Suppose that $\succcurlyeq$ satisfies $A_1 \succcurlyeq B_1 \succcurlyeq A_2 \succcurlyeq B_2 \succcurlyeq \ldots \succcurlyeq A_n \succcurlyeq B_n$ and $T_1 \succcurlyeq T_2 \succcurlyeq \ldots \succcurlyeq T_n$. Further suppose that we wish to check for consistency of $\mathcal{O}$ and do so using the trivial strategy (and hence we have a single context v with $\mathsf{core}_v = \top$). If in addition $\succcurlyeq$ satisfies $T_n \succcurlyeq A_1$ (i.e., the fresh predicates introduced during structural transformation are largest in the order), then our calculus will derive $2^{n+1} - 1$ clauses in $\mathcal{S}_v$. However, if instead $\succcurlyeq$ satisfies $A_1 \succcurlyeq T_n$ (i.e., the fresh predicates introduced during structural transformation are smallest in the order), then our calculus would instead only derive three clauses in $\mathcal{S}_v$ and then stop.

7.3 Clause Indexing Data Structures

As with first-order theorem provers, a crucial aspect of the implementation of Sequoia is the use of appropriate clause indexes—auxiliary data structures used to efficiently identify clauses required for an application of one of the inference rules of the calculus.

Sekar et al. [SRV01] and Graf [Gra96] present extensive overviews of the indexing techniques developed in the context of first-order theorem proving. While these techniques are known to give good performance, the restricted structure of DL-clauses and context clauses allow us to employ simpler, but still efficient, indexing techniques.

The remainder of this section is structured as follows. First, in Section 7.3.1, we detail how names, terms and literals are encoded using integers. Then, in Sections 7.3.2 to 7.3.4, we describe the indexing of context clauses for the Hyper, Pred, and Eq rules. Finally, in Section 7.3.5, we describe the data structure used to speed up forward redundancy checking and backward redundancy elimination.

7.3.1 Encoding Names, Terms and Literals using Integers

During the clausification phase (see Figure 7.1), each concept and role name in the input ontology is assigned a unique integer identifier (UID). Moreover, each term is also assigned a UID. These identifiers are assigned sequentially, which allows the use of perfect hashing in each index whose keys are names or terms. This representation also allows for an easy and efficient implementation of a context term order $\succ$ on a-terms from Definition 3 by directly comparing the assigned integers (but we must still be careful to satisfy condition (i) of the definition).

For compact representation, we represent each literal using a 64-bit integer that contains the type of the literal (either a unary atom, a binary atom, an equality, or an inequality), a flag specifying whether the literal is occurring in the body or the head of a clause, the name of the predicate for unary and binary atoms, and the terms occurring in the literal. In equalities and inequalities, terms are stored according to the term order—that is, $s \succ t$ holds in $s \approx t$ and $s \not\approx t$.

7.3.2 Unification Indexing for the Hyper Rule

To speed up the application of the Hyper rule, Sequoia maintains several indexes that can quickly identify the clauses that can participate in the inference rule. To facilitate indexing, we assign to each atom A a *unifier pattern* defined as follows

$$\mathsf{pattern}(A) = \begin{cases} B(x) & \text{if } A = B(x) \\ B(*) & \text{if } A = B(t) \text{ and } t \neq x \\ S(x,x) & \text{if } A = S(x,x) \\ S(x,*) & \text{if } A = S(x,t) \text{ and } t \neq x \\ S(*,x) & \text{if } A = S(t,x) \text{ and } t \neq x \end{cases}$$

where $*$ is a special wildcard symbol denoting any term. This definition exploits the restricted form of atoms in our setting and enjoys the following property: for all atoms A_1 and A_2 for which a substitution σ exists such that $A_1 = A_2\sigma$ and $\sigma(x) = x$ (as required by the preconditions of the Hyper rule), we have $\mathsf{pattern}(A_1) = \mathsf{pattern}(A_2)$.

During the ontology indexing phase, ontology $\mathcal{O}$ is indexed using two indexes. The index $\mathsf{ontologyIndex}_1$ is a hash table that maps a unifier pattern p to the set of all DL-clauses of $\mathcal{O}$ that

contain in their body an atom A whose unifier pattern is p:

$$\mathsf{ontologyIndex}_1[p] = \{\, \Gamma \to \Delta \in \mathcal{O} \mid \exists A \in \Gamma \text{ such that } p = \mathsf{pattern}(A) \,\}$$

Moreover, $\mathsf{ontologyIndex}_2$ is the set of all DL-clauses of $\mathcal{O}$ whose body is empty:

$$\mathsf{ontologyIndex}_2 = \{\, C \mid C = \Gamma \to \Delta \in \mathcal{O} \text{ with } \Gamma = \top \,\}$$

Finally, for each context $v \in \mathcal{V}$, we index $\mathcal{S}_v$ using a hash table $\mathsf{hyperIndex}_v$ that maps a unifier pattern p to the set of all clauses of $\mathcal{S}_v$ that contain in the head a maximal atom whose unifier pattern is p:

$$\mathsf{hyperIndex}_v[p] = \{\, \Gamma \to \Delta \in \mathcal{S}_v \mid \exists A \in \Delta \text{ such that } p = \mathsf{pattern}(A) \text{ and } \Delta \setminus \{A\} \not\succeq_v A \,\}$$

Indexes $\mathsf{ontologyIndex}_1$ and $\mathsf{ontologyIndex}_2$ contain the DL-clauses in $\mathcal{O}$ and are thus immutable, whereas index $\mathsf{hyperIndex}_v$ is updated whenever a clause is added to or removed from $\mathcal{S}_v$.

These indexes are used as follows. Assume that we wish to apply the Hyper rule to a head atom A of a clause C. Then, $\mathsf{ontologyIndex}_1[\mathsf{pattern}(A)]$ returns the set of clauses whose bodies contain an atom that unifies with A. For each such clause $\Gamma \to \Delta$ and each atom $A' \in \Gamma$, we query $\mathsf{hyperIndex}_v[\mathsf{pattern}(A')]$ to identify all the premises that can participate in the inference. Moreover, the clauses contained in $\mathsf{ontologyIndex}_2$ are added to each context as soon as the context is created.

7.3.3 Context Clause Indexing for the Pred Rule

To speed up the application of the Pred rule, for each context $v \in \mathcal{V}$, we index $\mathcal{S}_v$ using hash tables $\mathsf{predBodyIndex}_v$ and $\mathsf{predHeadIndex}_v$ that map an atom A to the sets of all clauses of $\mathcal{S}_v$ that contain A in the body and head, respectively:

$$\mathsf{predBodyIndex}_v[A] = \{\, \Gamma \to \Delta \in \mathcal{S}_v \mid A \in \Gamma \text{ and } \Delta \subseteq \mathsf{Pr}(\mathcal{O}) \,\}$$

$$\mathsf{predHeadIndex}_v[A] = \{\, \Gamma \to \Delta \in \mathcal{S}_v \mid A \in \Delta \text{ and } \Delta \setminus \{A\} \not\succeq_v A \,\}$$

These indexes are maintained whenever a clause is added to or removed from $\mathcal{S}_v$. To apply the Pred rule to a head atom A of a clause C, we query $\mathsf{predBodyIndex}_v[A']$ where A' is obtained from A in a way that enables unification; then, for each clause $\Gamma \to \Delta$ from this set and each atom $A'' \in \Gamma$, we query $\mathsf{predHeadIndex}_v[A''']$ where A''' is obtained from A''.

7.3.4 Context Clause Indexing for the Eq Rule

To speed up the application of the Eq rule, for each context $v \in \mathcal{V}$, we index $\mathcal{S}_v$ using the hash table $\mathsf{eqIndex}_v$ that maps a term s to the following set of all clauses of $\mathcal{S}_v$:

$$\begin{aligned}\mathsf{eqIndex}_v[s] = \{\, \Gamma \to \Delta \in \mathcal{S}_v \mid \exists L \in \Delta \text{ such that } \Delta \setminus \{L\} \not\succeq_v L \text{ and} \\ \text{(i)}\ \ L \text{ is an atom, } s \text{ is of the form } f(x), \text{ and } s \text{ occurs in } L, \text{ or} \\ \text{(ii)}\ \ L \text{ is of the form } s \approx t \text{ or } s \not\approx t, \text{ and } s \succ_v t \,\}\end{aligned}$$

Note that a term s in a head atom can participate in an inference rule application only if it is of the form $f(x)$, and so we only index such terms from atoms. This index is maintained whenever a clause is added to or removed from $\mathcal{S}_v$.

7.3.5 Redundancy Indexes

Redundancy indexes are used to speed up two related problems: given a clause C and a set of clauses N, *forward redundancy checking* determines whether N contains a clause that is stronger than C, and *backward redundancy elimination* retrieves all clauses of N that are weaker than C. (See Section 7.2.3 for the definition of when one clause is stronger or weaker than another.) The main difficulty in redundancy indexing is that the body and the head of a clause can match an arbitrary subset of the body and the head, respectively, of another clause, and that the matched literals need not be consecutive. There are exponentially many such subsets, which makes designing an efficient indexing structure difficult. For example, let $C_1 = \top \to A_2(x) \vee A_5(x)$ and let $C_2 = \top \to A_1(x) \vee A_2(x) \vee A_3(x) \vee A_4(x) \vee A_5(x) \vee A_6(x)$. Clearly, C_1 is stronger than C_2, but C_2 contains the disjunction $A_3(x) \vee A_4(x)$ between the atoms $A_2(x)$ and $A_5(x)$.

We have designed an index data structure inspired by *feature vector indexing* used in the theorem prover E [Sch13a], but devised specifically for context clauses. Unlike first-order theorem provers that must perform unification during redundancy checking, we can use so-called *per-*

$$C_1 = \top \to A_1(x) \vee A_2(x)$$
$$C_2 = \top \to A_2(x) \vee A_4(x) \vee A_5(x)$$
$$C_3 = \top \to A_6(x)$$
$$C_4 = \top \to A_2(x) \vee A_3(x)$$

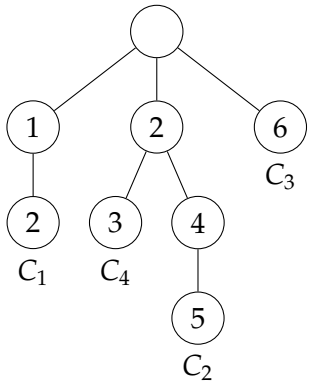


Figure 7.2: Example redundancy index for a set of context clauses. Each atom $A_i(x)$ is represented as the integer i.

fect indexing that returns exactly the required set of clauses. (By contrast, in first-order theorem provers, a feature vector index returns a set of clauses, and a full redundancy check must be subsequently performed on each clause in this set.)

We use a single index that supports both forward redundancy checking and backward redundancy elimination. To index a clause $C = \bigwedge_{i=1}^{m} L_i \to \bigvee_{i=m+1}^{n} L_i$, we construct a sequence of integers $\langle k_1, \ldots, k_n \rangle$ where k_i is the integer representation of L_i (see Section 7.3.1). We then sort this sequence into ascending order and use the sorted sequence as the key to insert the clause C into a trie data structure. Since we will always eliminate redundant clauses, the trie never contains distinct clauses $\Gamma_1 \to \Delta_1$ and $\Gamma_2 \to \Delta_2$ such that $\Gamma_1 \subseteq \Gamma_2$ and $\Delta_1 \subseteq \Delta_2$; consequently, clauses are always stored in the leaf nodes of the trie. Figure 7.2 shows an example set of context clauses $\{C_1, C_2, C_3, C_4\}$ and the corresponding redundancy index. In this example, for simplicity, we assume that each atom $A_i(x)$ is represented as the integer i.

We use one such index per context u, in which we index all clauses in $\mathcal{S}_u$ as well as all unprocessed clauses in context u; thus, our index maintains the set existing of all clauses in context u from Algorithm 1.

We now present algorithms that use the redundancy index to support forward redundancy checking and backward redundancy elimination. The algorithms use the following operations that take a sequence of integers $key = \langle k_1, \ldots, k_n \rangle$ as argument:

- ISEMPTY(key) returns *true* if key is the empty sequence (i.e., if $n = 0$);
- SUFFIXES(key) returns the set $\{\langle k_i, \ldots, k_n \rangle \mid 1 \leq i \leq n\}$;
- HEAD(key) returns the integer k_1;

Algorithm 2 Forward Redundancy Checking

```
1:  procedure CONTAINSSTRONGER(node, key)
2:      if ISLEAF(node) then
3:          return true
4:      else if ISEMPTY(key) then
5:          return false
6:      else
7:          for each subkey ∈ SUFFIXES(key) do
8:              child := CHILD(node, HEAD(subkey))
9:              if child ≠ null and CONTAINSSTRONGER(child, TAIL(subkey)) then
10:                 return true
11:         return false
```

- TAIL(key) returns the sequence $\langle k_2, \ldots, k_n \rangle$.

Moreover, the algorithms also use the following operations that take a trie node as argument:

- ISLEAF$(node)$ returns *true* if *node* is a leaf node;
- CHILD$(node, k)$ returns the child of *node* labelled with literal integer representation k, or *null* if no such child exists;
- CHILDREN$(node)$ returns the set of all children of *node*;
- LITERAL$(node)$ returns the literal integer representation labelled at *node*;
- CLAUSE$(node)$ returns the clause associated with *node* (assuming the latter is a leaf).

Forward Redundancy Checking To check whether the redundancy index contains a clause that is stronger than a clause C, we construct the unique integer sequence *key* corresponding to C as described earlier, and we search the trie by calling CONTAINSSTRONGER$(root, key)$ as described in Algorithm 2 where *root* is the root of the trie. The crux of the algorithm is in line 7, where, given a sequence $key = \langle k_1, \ldots, k_n \rangle$, the algorithm attempts to find, for each k_i with $1 \leq i \leq n$, a child of *node* labelled with k_i, thus skipping over gaps in the clause.

Backward Redundancy Elimination To retrieve from the redundancy index all clauses that are stronger than a clause C, we construct the unique integer sequence *key* corresponding to C as described earlier, and we search the trie by calling SELECTWEAKER$(root, key)$ as described in Algorithm 3 where *root* is the root of the trie. If the procedure is called with *node* and an empty sequence, in lines 3 to 8 it collects all clauses underneath *node*. Otherwise, in lines 9 to 14 the algorithm examines each child of *node*. If the child is labelled with a literal integer representation

Algorithm 3 Backward Redundancy Elimination

```
 1: procedure SELECTWEAKER(node, key)
 2:     R := ∅
 3:     if ISEMPTY(key) then
 4:         if ISLEAF(node) then
 5:             R := {CLAUSE(node)}
 6:         else
 7:             for each child ∈ CHILDREN(node) do
 8:                 R := R ∪ SELECTWEAKER(child, key)
 9:     else
10:         for each child ∈ CHILDREN(node) do
11:             if LITERAL(child) < HEAD(key) then
12:                 R := R ∪ SELECTWEAKER(child, key)
13:             else if LITERAL(child) = HEAD(key) then
14:                 R := R ∪ SELECTWEAKER(child, TAIL(key))
15:     return R
```

larger than the first element of *key*, then no clause in the subtree under *node* is weaker than C. If the child is labelled with a literal integer representation smaller than the first element of *key*, then in line 12 the algorithm tries to match *key* in the subtree under *node*. Finally, if the child is labelled with a literal integer representation equal to the first element of *key*, then in line 14 the algorithm tries to match the rest of *key* in the subtree under *node*.

Chapter 8

Evaluation

In this chapter, we describe an evaluation of Sequoia over a well-known ontology corpus, and we compare Sequoia to established DL reasoners for the time taken to compute an ontology classification. Furthermore, during the development of Sequoia, we observed that redundancy elimination was one of the main bottlenecks of the system, and thus we also evaluate the effectiveness of our redundancy indexes on performance.

8.1 Corpus and Experiment Set-Up

In this section, we describe the corpus and testing environment used for the evaluation. We used the Oxford Ontology Repository[1] from which we excluded 7 ontologies with irregular RBoxes. Since Sequoia does not support datatypes or nominals, we made the following modifications to some of the test ontologies to reduce their expressivity to $\mathcal{SRIQ}$:

(i) we systematically replaced each occurrence of a particular data property with a fresh object property, and replaced all datatype related axioms with their corresponding axioms for object properties; and

(ii) we approximated each nominal by replacing it with a fresh class in the usual way [SCP06], where every occurrence of a particular nominal was replaced by the same fresh class.

Furthermore, we removed ABox assertions. After this we removed 112 ontologies that did not contain any classes (for example, ontologies that only contained ABox data). We thus obtained a

[1] https://www.cs.ox.ac.uk/isg/ontologies/

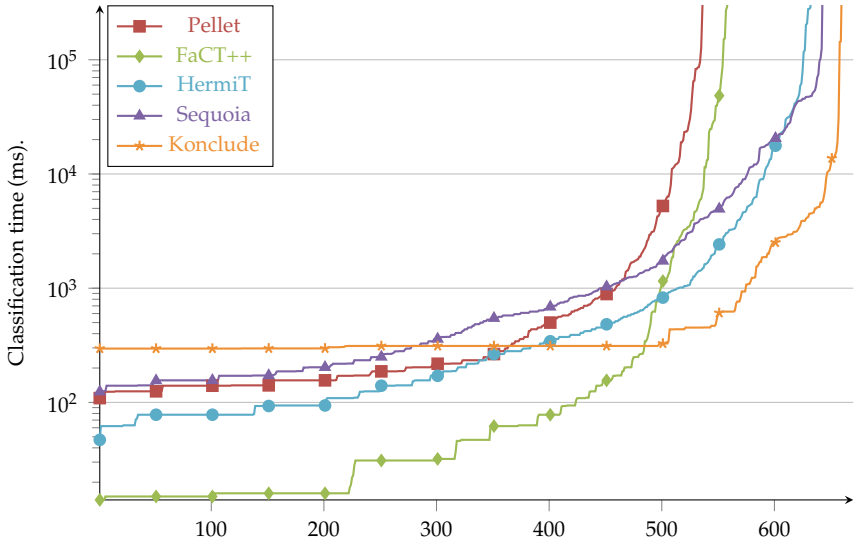


Figure 8.1: Classification times for all ontologies

corpus of 665 ontologies on which we tested all reasoners.

We ran our experiments on a Dell workstation with two Intel Xeon E5-2643 v3 3.4 GHz processors with 6 cores per processor and 128 GB of RAM running Windows Server 2012 R2. We used Java 8 update 66 with 15 GB of heap memory allocated to each Java reasoner, and a maximum private working set size of 15 GB for each reasoner in native code. In each test, we measured the wall-clock classification time; this excludes parsing time for reasoners based on the OWL API (i.e., HermiT, Pellet, FaCT++, and Sequoia). Each test was given a timeout of 5 minutes. We report the average time over three runs, unless an exception or timeout occurred in one of the three runs, in which case we report failure.

8.2 Evaluation of Sequoia and Comparison to Other Reasoners

In this section, we compare Sequoia to existing DL reasoners. We have evaluated our system using the methodology by Steigmiller et al. [SLG14] by comparing Sequoia with HermiT 1.3.8, Pellet 2.3.1, FaCT++ 1.6.4, and Konclude 0.6.2. Unlike the other systems, Konclude supports reasoning using multiple threads; hence, we used Konclude in single-threaded mode in order to compare the underlying calculi. Moreover, Sequoia was configured to use the cautious strategy (since this causes our calculus to mimic existing calculi on $\mathcal{ELH}$ ontologies as described in Section 5.3). All systems, ontologies, and test results are available online.[2]

[2] https://krr-nas.cs.ox.ac.uk/2016/Sequoia/

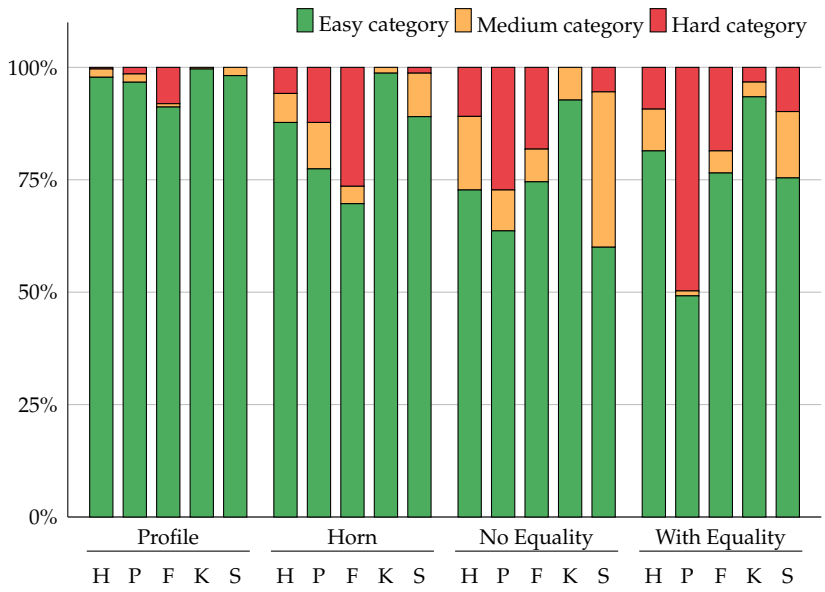


Figure 8.2: Percentage of easy, medium and hard ontologies per ontology group for HermiT (H), Pellet (P), FaCT++ (F), Konclude (K), and Sequoia (S)

Figure 8.1 shows an overview of the classification times for the entire corpus. The y-axis shows the classification times on a logarithmic scale, and timeouts are shown as infinity. A number n on the x-axis represents the n-th easiest ontology for a reasoner with ontologies sorted (for that reasoner) in the ascending order of classification time. For example, a point (50, 100) on a curve for a particular reasoner means that the 50th easiest ontology for that reasoner took 100 ms to classify.

Sequoia could classify most ontologies (576 out of 665) in under 10s, which is consistent with the other reasoners. Sequoia was fairly robust, failing on only 23 ontologies; in contrast, HermiT failed on 33, Pellet on 129, FaCT++ on 107, and Konclude on 6 ontologies. Moreover, Sequoia succeeded on 14 ontologies on which all of HermiT, Pellet and FaCT++ failed. Finally, there was one ontology where Sequoia succeeded and all other reasoners failed; this was a hard version of FMA (ID 00285) that uses both disjunctions and cardinality constraints.

Figure 8.2 shows an overview of how each reasoner performed on different types of ontology. We partitioned the ontologies into the following four groups: within a profile of OWL 2 DL (i.e., captured by OWL 2 EL, QL, or RL); Horn but not in a profile; disjunctive but without cardinality constraints; and disjunctive and with cardinality constraints. We used the OWL API to determine profile membership, and we identified the remaining three groups after structural transformation. In addition, for each reasoner, we categorise each ontology as either *easy* ($<$ 10s),

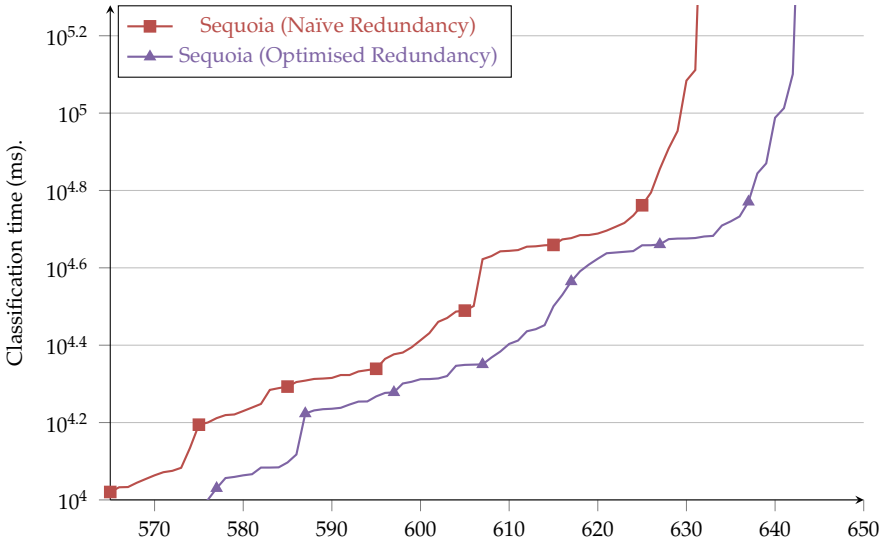


Figure 8.3: Effects of redundancy indexing on classification times for medium ontologies

medium (10s to 5min), or *hard* (timeout or exception). The figure depicts a bar for each reasoner and group, where each bar is divided into blocks representing the percentage of ontologies in each of the aforementioned categories of difficulty. For Sequoia, over 98% of in-profile ontologies and over 91% of out-of-profile Horn ontologies are easy, with the remainder being of medium difficulty. Sequoia timed out largely on ontologies containing both disjunctions and cardinality constraints, and even in this case only Konclude timed out in fewer cases.

In summary, although only an early prototype, Sequoia is a competitive reasoner that comfortably outperforms HermiT, Pellet, and FaCT++, and which exhibits nice pay-as-you-go behaviour. Finally, problematic ontologies seem to mostly contain complex RBoxes or large numbers in cardinality constraints, which suggests promising directions for future optimisation.

8.3 Redundancy Indexing

In this section, we evaluate the effectiveness of the index used for forward redundancy checking and backward redundancy elimination that we described in Section 7.3.5. In our experience of developing Sequoia, these related tasks were the main performance bottlenecks of the system. To evaluate the effectiveness of our redundancy indexing data structure, we have rerun the same experiments as described in the previous section to compare Sequoia with a version in which a naïve data structure for redundancy indexing was implemented. The results for the ontologies whose classification time exceeded 10s when using the naïve data structure are depicted in

Figure 8.3. Sequoia is consistently faster with the optimised index. The difference is typically very significant, and in comparison to the naïve data structure for redundancy indexing, our optimised index allowed us to classify 11 additional ontologies within the time limit.

Chapter 9

Conclusion

In this thesis, we have presented the first consequence-based calculus for $\mathcal{SRIQ}$, a description logic that includes both disjunction and cardinality constraints. $\mathcal{SRIQ}$ is closely related to the ontology language OWL 2 DL, whose logical underpinning is $\mathcal{SROIQ}(\mathbf{D})$, which extends $\mathcal{SRIQ}$ with nominals and datatypes.

Our calculus combines ideas from state-of-the-art first-order superposition and hypertableau calculi, including the use of ordered paramodulation for equality reasoning. A key difference between our consequence-based procedure for $\mathcal{SRIQ}$ and both first-order superposition and hypertableau calculi is that our calculus is not just refutation complete: the required consequences are derived during saturation. Furthermore, with the standard assumption that numbers in cardinality constraints are encoded in unary, our algorithms for ontology classification and ontology satisfiability testing run in exponential time in the size of the ontology. Hence, our algorithms are worst-case optimal. Despite the increased complexity of $\mathcal{SRIQ}$, the calculus mimics existing calculi on $\mathcal{ELH}$ ontologies.

To demonstrate the practicality of our approach, we implemented our calculus in a new reasoning called Sequoia. Our system incorporates a number of important indexing data structures and optimisation techniques. The performance of Sequoia was evaluated on a corpus of ontologies and compared against other state-of-the-art reasoning systems. The performance of individual indexing strategies and optimisations were also evaluated. Although Sequoia does not match the performance of the best-performing system, the results show that Sequoia is competitive with well-established reasoners and it exhibits nice pay-as-you-go behaviour in practice.

We believe that further research into improvements to the calculus and the development of additional optimisation techniques will lead to a very competitive and robust system, and that the calculus presented in this thesis provides the theoretical foundation upon which subsequent calculi can be based.

9.1 Future Work

The results obtained in this thesis suggest a few key areas for future research, which we will discuss next.

Support for Datatypes The DL $\mathcal{SRIQ}(\mathbf{D})$ is the extension of $\mathcal{SRIQ}$ with datatypes—a simplified variant of concrete domains [BH91]—which allow for reasoning with concrete data, such as strings or integers.

Unfortunately, it has been shown that a logic with general inclusion axioms and concrete domains is undecidable [Lut04]. Consequently, several restrictions of the general approach have been investigated in the literature [HS01, PH02, HMW01]; a survey of the main results is presented by Lutz [Lut03]. The cumulative results of this research have influenced the design of OWL 2 [MPP12], which supports datatypes—a basic form of concrete domains.

It would therefore be both interesting and useful to extend the reasoning capability of our calculus from $\mathcal{SRIQ}$ to $\mathcal{SRIQ}(\mathbf{D})$. Such a calculus would thus be able to handle all of OWL 2 DL except nominals. By following the ideas presented by Motik [Mot06], we are confident that the calculus could be extended to support datatypes.

The basic ideas for supporting such reasoning within the framework of the current calculus are as follows. Firstly, a concrete domain must be introduced in addition to the Herbrand universe such that each datatype is given an interpretation in the concrete domain. For example, the datatype `int` would be mapped to the subset of the concrete domain that contains precisely all the integers. Secondly, a new distinct sort c of function symbols, called the *concrete sort*, would be introduced to distinguish terms related to datatype reasoning, and only terms of the concrete sort c would be allowed as subterms of datatype atoms. For example, if $g(x)$ is a term of sort c, and `int` is a function symbol with symbol type $\mathsf{c} \to \mathsf{p}$, then the clause $\top \to \mathtt{int}(g(x))$ would represent that there exists a g-successor that should be assigned an integer element in any model.

Thirdly, equalities and inequalities between concrete terms would allow for the expression of cardinality constraints. For example, if $g_1(x)$ and $g_2(x)$ are both terms of sort c, then the clause $\top \to \mathsf{INEQ}(g_1(x), g_2(x))$ would represent that the g_1-successor and the g_2-successor should be assigned different concrete elements in any model, where INEQ is a special predicate encoded using a function symbol in the usual way.

Support for Nominals, ABox Reasoning, and SPARQL Adding support for nominals, together with datatype support, would allow our calculus to handle all features of OWL 2 DL. Furthermore, since the ABox provides a restricted form of reasoning with nominals, this would allow for ABox reasoning.

The SPARQL query language [HS13] is a standard query language for semantic reasoning systems. SPARQL 1.1 includes several entailment regimes [GO13, GK10] that support querying the implicit information that is logically entailed by the ontology. The OWL 2 Direct Semantics Entailment Regime has been specifically designed to allow for query evaluation to be implemented using the standard reasoning services provided by an OWL 2 reasoner. Hence, given ABox reasoning support, it would be possible to implement SPARQL support for Sequoia using the known techniques [KG13, GHMS+14].

However, in contrast to datatype reasoning, handling ABox reasoning and nominals seems to be much more involved. In fact, adding nominals to $\mathcal{ALCHIQ}^+$ raises the complexity of reasoning to NEXPTIME [Tob01] so a worst-case optimal calculus must be nondeterministic, which is quite different from all consequence-based calculi of which we are aware. Furthermore, although a superposition-based calculus for handling $\mathcal{SHOIQ}$ was proposed by Kazakov and Motik [KM08b], the technique presented is not worst-case optimal.

Improved Handling for Cardinality Constraints In our approach, cardinality constraints are encoded into equations as shown in Table 2.3. Unfortunately, this transformation results in a quadratic blowup in the number of clauses when encoding at-least restrictions, and a quadratic blowup in the number of equalities when encoding at-most restrictions. Furthermore, the number of equalities and inequalities that are generated for cardinality constraints, even when relatively small numbers are used in the constructors, has been shown in practice to result in a large number of redundant clauses generated by the Eq rule.

It may be possible to further restrict the inferences performed by the Eq rule at the expense of weakening the ordering restrictions, akin to the *basic restriction* in paramodulation calculi [NR01]. Roughly, the idea of basic paramodulation is to disallow paramodulation inferences into terms introduced by substitutions from a previous inference step. In our calculus, this would include terms introduced during unification performed by the Hyper rule. However, to retain completeness, paramodulation is instead allowed to take place on *both* sides of the equation containing the maximal term of an equality (instead of only on the maximal side of the equation containing the maximal term). Note, however, that this relaxation of the ordering restriction does not affect paramodulation with equations that encode non-equality predicates.

Basic superposition was used in the reasoning algorithms proposed by Motik [Mot06] and Kazakov and Motik [KM08a], however, there it was required to prevent unbounded term nesting which may otherwise lead to non-termination. Furthermore, our calculus already incorporates some notion of a basic restriction for paramodulation, since the nesting of function symbols of sort a is disallowed by the syntactic form of context clauses, and the only equality that can be propagated to a predecessor by the Pred rule is $x \approx y$. Experimental evaluation of the basic restriction in first-order theorem proving has shown that this restriction may lead to the generation of fewer clauses [BGLS92], but additional evaluation of its merit in our setting (where term depth nesting is already restricted) is required.

Moreover, a further challenge is to modify the calculus so that it can effectively deal with large numbers in cardinality constraints. One approach that has demonstrated robust performance in such cases is to combine tableau calculi with algebraic methods, namely integer programming, instead of using don't-know nondeterminism to solve algebraic reasoning by covering all possible cases [PH12]. While a combination of saturation-based calculi and algebraic methods has been previously proposed for $\mathcal{ELQ}$ by Vlasenko et al. [VDHJ16], it is not clear how to extend this combination to a logic that admits both cardinality constraints and disjunction.

Bibliography

[ABBB+00] Michael Ashburner et al. Gene Ontology: tool for the unification of biology. *Nature Genetics*, 25(1):25–29, 2000 (cited on pages 1, 25).

[ACH12] Ana Armas Romero, Bernardo Cuenca Grau, and Ian Horrocks. MORe: Modular Combination of OWL Reasoners for Ontology Classification. In *Proceedings of the 11th International Semantic Web Conference (ISWC 2012)*, volume 7649 of *Lecture Notes in Computer Science*, pages 1–16. Springer, 2012 (cited on pages 5, 32).

[ANvB98] Hajnal Andréka, István Németi, and Johan van Benthem. Modal Languages and Bounded Fragments of Predicate Logic. *Journal of Philosophical Logic*, 27(3):217–274, 1998 (cited on pages 2, 24).

[Baa03] Franz Baader. Terminological Cycles in a Description Logic with Existential Restrictions. In Georg Gottlob and Toby Walsh, editors, *Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI 2003)*, pages 325–330. Morgan-Kaufmann, 2003 (cited on pages 25, 30).

[BBH96] Franz Baader, Martin Buchheit, and Bernhard Hollander. Cardinality restrictions on concepts. *Artificial Intelligence*, 88(1):195–213, 1996 (cited on page 26).

[BBL05] Franz Baader, Sebastian Brandt, and Carsten Lutz. Pushing the $\mathcal{EL}$ Envelope. In Leslie Pack Kaelbling and Alessandro Saffiotti, editors, *Proceedings of the 19th International Joint Conference on Artificial Intelligence (IJCAI 2005)*, IJCAI 2005, pages 364–369. Morgan Kaufmann, 2005 (cited on pages 4, 19, 25, 30, 31, 38, 39, 57, 85–87).

[BCMN+10] Franz Baader, Diego Calvanese, Deborah McGuinness, Daniele Nardi, and Peter F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation*

and Applications. Cambridge University Press, 2nd edition, 2010 (cited on pages 2, 18, 21, 57).

[BDS93] Martin Buchheit, Francesco M. Donini, and Andrea Schaerf. Decidable Reasoning in Terminological Knowledge Representation Systems. *Journal of Artificial Intelligence Research*, 1(1):109–138, 1993 (cited on pages 26, 38).

[BG01] Leo Bachmair and Harald Ganzinger. Resolution Theorem Proving. In John Alan Robinson and Andrei Voronkov, editors, *Handbook of Automated Reasoning*. Volume I, chapter 2, pages 19–99. Elsevier Science, 2001 (cited on pages 29, 38).

[BGLS92] Leo Bachmair, Harald Ganzinger, Christopher Lynch, and Wayne Snyder. Basic Paramodulation and Superposition. In Deepak Kapur, editor, *Proceedings of the 11th International Conference on Automated Deduction (CADE-11)*, volume 607 of *Lecture Notes in Computer Science*, pages 462–476. Springer, 1992 (cited on page 114).

[BH91] Franz Baader and Philipp Hanschke. A Scheme for Integrating Concrete Domains into Concept Languages. In *Proceedings of the 12th International Joint Conference on Artificial Intelligence (IJCAI 1991)*, pages 452–457. Morgan Kaufmann, 1991 (cited on page 112).

[BHNP+94] Franz Baader, Bernhard Hollunder, Bernhard Nebel, Hans-Jürgen Profitlich, and Enrico Franconi. An empirical analysis of optimization techniques for terminological representation systems. *Applied Intelligence*, 4(2):109–132, 1994 (cited on page 27).

[BL84] Ronald J. Brachman and Hector J. Levesque. The Tractability of Subsumption in Frame-Based Description Languages. In Ronald J. Brachman, editor, *Proceedings of the 4th National Conference on Artificial Intelligence (AAAI-84)*, pages 34–37. AAAI Press, 1984 (cited on page 23).

[BLS05] Franz Baader, Carsten Lutz, and Boontawee Suntisrivaraporn. Is Tractable Reasoning in Extensions of the Description Logic $\mathcal{EL}$ Useful in Practice? In *Proceedings of the 4th International Workshop on Methods for Modalities (M4M-4)*, 2005 (cited on pages 25, 31).

[BLS06] Franz Baader, Carsten Lutz, and Boontawee Suntisrivaraporn. CEL—A Polynomial-time Reasoner for Life Science Ontologies. In Ulrich Furbach and Natarajan Shankar, editors, *Proceedings of the 3rd International Joint Conference on Automated Reasoning (IJCAR 2006)*, volume 4130 of *LNCS*, pages 287–291. Springer, 2006 (cited on page 4).

[BN98] Franz Baader and Tobias Nipkow. *Term Rewriting and All That*. Cambridge University Press, 1998 (cited on pages 14, 15, 49, 81).

[Bor96] Alex Borgida. On the relative expressiveness of description logics and predicate logics. *Artificial Intelligence*, 82(1-2):353–367, 1996 (cited on page 2).

[BS06] Peter Baumgartner and Renate A. Schmidt. Blocking and Other Enhancements for Bottom-Up Model Generation Methods. In Ulrich Furbach and Natarajan Shankar, editors, *Proceedings of the 3rd International Joint Conference on Automated Reasoning (IJCAR 2006)*, volume 4130 of *LNCS*, pages 125–139. Springer, 2006 (cited on page 26).

[CCKK+15] Diego Calvanese, Benjamin Cogrel, Elem Guzel Kalayci, Sarah Komla-Ebri, Roman Kontchakov, Davide Lanti, Martin Rezk, Mariano Rodriguez-Muro, and Guohui Xiao. OBDA with the Ontop Framework. In Domenico Lembo, Riccardo Torlone, and Andrea Marrella, editors, *Proceedings of the 23rd Italian Symposium on Advanced Database Systems (SEBD 2015)*, pages 296–303. Curran Associates, Inc., 2015 (cited on page 2).

[CHMP+08] Bernardo Cuenca Grau, Ian Horrocks, Boris Motik, Bijan Parsia, Peter Patel-Schneider, and Ulrike Sattler. OWL 2: The next step for OWL. *Journal of Web Semantics: Science, Services and Agents on the World Wide Web*, 6(4):309–322, 2008 (cited on page 1).

[DdN05] Stéphane Demri and Hans de Nivelle. Deciding Regular Grammar Logics with Converse Through First-Order Logic. *Journal of Logic, Language and Information*, 14(3):289–329, 2005 (cited on page 20).

[DLNS98] Francesco M. Donini, Maurizio Lenzerini, Daniele Nardi, and Andrea Schaerf. AL-log: Integrating Datalog and Description Logics. *Journal of Intelligent Information Systems*, 10(3):227–252, 1998 (cited on page 38).

[dNdR03] Hans de Nivelle and Maarten de Rijke. Deciding the Guarded Fragments by Resolution. *Journal of Symbolic Computation*, 35(1):21–58, 2003 (cited on page 28).

[dNP01] Hans de Nivelle and Ian Pratt-Hartmann. A Resolution-Based Decision Procedure for the Two-Variable Fragment with Equality. In *Proceedings of the 1st International Joint Conference on Automated Reasoning (IJCAR 2001)*, pages 211–225. Springer-Verlag, 2001 (cited on page 28).

[dNSH00] Hans de Nivelle, Renate A. Schmidt, and Ullrich Hustadt. Resolution-Based Methods for Modal Logics. *Logic Journal of the IGPL*, 8(3):265–292, 2000 (cited on pages 28, 39).

[Don10] Francesco M. Donini. Complexity of Reasoning. In Franz Baader, Diego Calvanese, Deborah McGuinness, Daniele Nardi, and Peter F. Patel-Schneider, editors, *The Description Logic Handbook: Theory, Implementation and Applications*, chapter 3, pages 105–148. Cambridge University Press, 2nd edition, 2010 (cited on page 27).

[DRP07] Sebastian Derriere, André Richard, and Andrea Preite-Martinez. An Ontology of Astronomical Object Types for the Virtual Observatory. *Highlights of Astronomy*, 14:603–603, 2007 (cited on page 1).

[FLHT01] Christian G. Fermüller, Alexander Leitsch, Ullrich Hustadt, and Tanel Tammet. Resolution Decision Procedures. In John Alan Robinson and Andrei Voronkov, editors, *Handbook of Automated Reasoning*. Volume II, chapter 25, pages 1791–1849. Elsevier Science, 2001 (cited on page 29).

[GBJM+13] Rafael S. Gonçalves, Samantha Bail, Ernesto Jiménez-Ruiz, Nicolas Matentzoglu, Bijan Parsia, Birte Glimm, and Yevgeny Kazakov. OWL Reasoner Evaluation (ORE) Workshop 2013 Results: Short Report. In Samantha Bail, Birte Glimm, Rafael S. Gonçalves, Ernesto Jiménez-Ruiz, Yevgeny Kazakov, Nicolas Matentzoglu, and Bijan Parsia, editors, *Informal Proceedings of the 2nd International*

Workshop on OWL Reasoner Evaluation (ORE 2013), volume 1015 of *CEUR Workshop Proceedings*, pages 1–18. CEUR-WS.org, 2013 (cited on page 31).

[GdN99] Harald Ganzinger and Hans de Nivelle. A Superposition Decision Procedure for the Guarded Fragment with Equality. In *Proceedings of the 14th Annual IEEE Symposium on Logic in Computer Science*, pages 295–303. IEEE Computer Society, 1999 (cited on pages 28, 39, 60).

[GHM10] Birte Glimm, Ian Horrocks, and Boris Motik. Optimized Description Logic Reasoning via Core Blocking. In Jürgen Giesl and Reiner Hähnle, editors, *Proceedings of the 5th International Joint Conference on Automated Reasoning (IJCAR 2010)*, volume 6173 of *Lecture Notes in Artificial Intelligence*, pages 457–471. Springer, 2010 (cited on page 28).

[GHMS+12] Birte Glimm, Ian Horrocks, Boris Motik, Rob Shearer, and Giorgos Stoilos. A Novel Approach to Ontology Classification. *Journal of Web Semantics: Science, Services and Agents on the World Wide Web*, 14:84–101, 2012 (cited on pages 27, 30).

[GHMS+14] Birte Glimm, Ian Horrocks, Boris Motik, Giorgos Stoilos, and Zhe Wang. HermiT: An OWL 2 Reasoner. *Journal of Automated Reasoning*, 53(3):245–269, 2014 (cited on page 113).

[GHS03] Lilia Georgieva, Ullrich Hustadt, and Renate A. Schmidt. Hyperresolution for Guarded Formulae. *Journal of Symbolic Computation*, 36(1–2):163–192, 2003 (cited on page 39).

[GK10] Birte Glimm and Markus Krötzsch. SPARQL Beyond Subgraph Matching. In Peter F. Patel-Schneider, Yue Pan, Birte Glimm, Pascal Hitzler, Peter Mika, Jeff Pan, and Ian Horrocks, editors, *Proceedings of the 9th International Semantic Web Conference (ISWC 2010)*, volume 6496 of *LNCS*, pages 241–256. Springer, 2010 (cited on page 113).

[GN07] Rajeev Goré and Linh Anh Nguyen. EXPTIME Tableaux with Global Caching for Description Logics with Transitive Roles, Inverse Roles and Role Hierarchies. In Nicola Olivetti, editor, *Proceedings of the 16th International Conference on Au-*

tomated Reasoning with Analytic Tableaux and Related Methods (TABLEAUX 2007), volume 4548 of *LNCS*, pages 133–148. Springer, 2007 (cited on pages 28, 38).

[GO13] Birte Glimm and Chimezie Ogbuji. SPARQL 1.1 Entailment Regimes, W3C Recommendation. `https://www.w3.org/TR/sparql11-entailment/`, 2013 (cited on page 113).

[Goo05] John Goodwin. Experiences of Using OWL at the Ordnance Survey. In *Proceedings of the 1st International Workshop on OWL Experiences and Directions (OWLED 2005)*, volume 188 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2005 (cited on page 1).

[GOR97] Erich Grädel, Martin Otto, and Eric Rosen. Two-Variable Logic with Counting is Decidable. In *Proceedings of the 12th IEEE Symposium on Logic in Computer Science (LICS 1997)*, pages 306–317. IEEE Computer Society, 1997 (cited on pages 2, 24).

[Gra96] Peter Graf. *Term Indexing*, volume 1053 of *Lecture Notes in Computer Science*. Springer, 1996 (cited on page 97).

[Gru93] Thomas R. Gruber. A Translation Approach to Portable Ontology Specifications. *Knowledge Acquisition*, 5(2):199–220, 1993 (cited on page 1).

[GZB06] Christine Golbreich, Songmao Zhang, and Olivier Bodenreider. The Foundational Model of Anatomy in OWL: Experience and Perspectives. *Journal of Web Semantics: Science, Services and Agents on the World Wide Web*:181–195, 2006 (cited on page 1).

[HB11] Matthew Horridge and Sean Bechhofer. The OWL API: A Java API for OWL Ontologies. *Semantic Web*, 2(1):11–21, 2011 (cited on page 91).

[HG08] Ian Horrocks and Henson Graves. Application of OWL 1.1 to Systems Engineering. In Kendall Clark and Peter F. Patel-Schneider, editors, *Proceedings of the 4th International Workshop on OWL Experiences and Directions (OWLED 2008)*, volume 496 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2008 (cited on page 1).

[HKS05] Ian Horrocks, Oliver Kutz, and Ulrike Sattler. The Irresistible $\mathcal{SRIQ}$. In *Proceedings of the 1st International Workshop on OWL Experiences and Directions (OWLED 2005)*, 2005 (cited on page 5).

[HKS06] Ian Horrocks, Oliver Kutz, and Ulrike Sattler. The Even More Irresistible $\mathcal{SROIQ}$. In *Proceedings of the 10th International Conference on Principles of Knowledge Representation and Reasoning (KR 2006)*, pages 57–67. AAAI Press, 2006 (cited on pages 15, 20, 24, 91).

[HM01] Volker Haarslev and Ralf Möller. RACER System Description. In Rajeev Goré, Alexander Leitsch, and Tobias Nipkow, editors, *Proceedings of the 1st International Joint Conference on Automated Reasoning (IJCAR 2001)*, volume 2083 of *Lecture Notes in Computer Science*, pages 701–705. Springer-Verlag, 2001 (cited on pages 3, 24, 27).

[HMS04] Ullrich Hustadt, Boris Motik, and Ulrike Sattler. Reducing $\mathcal{SHIQ}^-$ Description Logic to Disjunctive Datalog Programs. In Didier Dubois, Christopher A. Welty, and Mary-Anne Williams, editors, *Proceedings of the 9th International Conference on Principles of Knowledge Representation and Reasoning (KR 2004)*, pages 152–162. AAAI Press, 2004 (cited on page 30).

[HMS08] Ullrich Hustadt, Boris Motik, and Ulrike Sattler. Deciding Expressive Description Logics in the Framework of Resolution. *Information and Computation*, 206(5):579–601, 2008 (cited on pages 28, 29, 39).

[HMW01] Volker Haarslev, Ralf Möller, and Michael Wessel. The Description Logic $\mathcal{ALCNH}_{R^+}$ Extended with Concrete Domains: A Practically Motivated Approach. In *Proceedings of the 1st International Joint Conference on Automated Reasoning (IJCAR 2001)*, pages 29–44. Springer-Verlag, 2001 (cited on page 112).

[Hor10] Ian Horrocks. Implementation and Optimisation Techniques. In Franz Baader, Diego Calvanese, Deborah McGuinness, Daniele Nardi, and Peter F. Patel-Schneider, editors, *The Description Logic Handbook: Theory, Implementation and Applications*, chapter 9, pages 306–346. Cambridge University Press, 2nd edition, 2010 (cited on page 26).

[Hor11] Ian Horrocks. Tool Support for Ontology Engineering. In Dieter Fensel, editor, *Foundations for the Web of Information and Services*, pages 103–112. Springer, 2011 (cited on page 2).

[Hor97] Ian Horrocks. *Optimising Tableaux Decision Procedures for Description Logics*. PhD thesis, University of Manchester, UK, 1997 (cited on page 24).

[Hor98] Ian Horrocks. Using an Expressive Description Logic: FaCT or Fiction? In *Proceedings of the 6th International Conference on Principles of Knowledge Representation and Reasoning (KR 1998)*, pages 636–647, 1998 (cited on pages 24, 26).

[HS01] Ian Horrocks and Ulrike Sattler. Ontology Reasoning in the $\mathcal{SHOQ}(D)$ Description Logic. In *Proceedings of the 17th International Joint Conference on Artificial Intelligence (IJCAI 2001)*, pages 199–204. Morgan Kaufmann, 2001 (cited on page 112).

[HS04] Ian Horrocks and Ulrike Sattler. Decidability of $\mathcal{SHIQ}$ with Complex Role Inclusion Axioms. *Artificial Intelligence*, 160(1–2):79–104, 2004 (cited on pages 19, 20, 24).

[HS07] Ian Horrocks and Ulrike Sattler. A Tableau Decision Procedure for $\mathcal{SHOIQ}$. *Journal of Automated Reasoning*, 39(3):249–276, 2007 (cited on pages 24, 26).

[HS13] Steven Harris and Andy Seaborne. SPARQL 1.1 Query Language, W3C Recommendation. `https://www.w3.org/TR/sparql11-query/`, 2013 (cited on page 113).

[HS99a] Ian Horrocks and Ulrike Sattler. A Description Logic with Transitive and Inverse Roles and Role Hierarchies. *Journal of Logic and Computation*, 9(3):385–410, 1999 (cited on page 28).

[HS99b] Ullrich Hustadt and Renate A. Schmidt. On the Relation of Resolution and Tableaux Proof Systems for Description Logics. In *Proceedings of the 16th International Joint Conference on Artificial Intelligence (IJCAI 1999)*, pages 110–117, 1999 (cited on page 28).

[HST00a] Ian Horrocks, Ulrike Sattler, and Stephan Tobies. Practical Reasoning for Very Expressive Description Logics. *Logic Journal of the IGPL*, 8(3):239–264, 2000 (cited on pages 19, 24, 60).

[HST00b] Ian Horrocks, Ulrike Sattler, and Stephan Tobies. Reasoning with Individuals for the Description Logic $\mathcal{SHIQ}$. In David McAllester, editor, *Proceedings of the 17th International Conference on Automated Deduction (CADE-17)*, volume 1831 of *Lecture Notes in Computer Science*, pages 482–496. Springer, 2000 (cited on page 27).

[HW06] Alexander K. Hudek and Grant E. Weddell. Binary Absorption in Tableaux-Based Reasoning for Description Logics. In Bijan Parsia, Ulrike Sattler, and David Toman, editors, *Proceedings of the 19th International Workshop on Description Logics (DL 2006)*, volume 189 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2006 (cited on page 26).

[Joy76] William H. Joyner Jr. Resolution Strategies As Decision Procedures. *Journal of the ACM*, 23(3):398–417, 1976 (cited on page 29).

[Kaz08] Yevgeny Kazakov. $\mathcal{RIQ}$ and $\mathcal{SROIQ}$ are Harder than $\mathcal{SHOIQ}$. In Gerhard Brewka and Jérôme Lang, editors, *Proceedings of the 11th International Conference on Principles of Knowledge Representation and Reasoning (KR 2008)*, pages 274–284. AAAI Press, 2008 (cited on pages 3, 5, 19, 29).

[Kaz09] Yevgeny Kazakov. Consequence-Driven Reasoning for Horn $\mathcal{SHIQ}$ Ontologies. In Craig Boutilier, editor, *Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI 2009)*, pages 2040–2045. IJCAI, 2009 (cited on pages 4, 28, 30, 31, 39, 60).

[KdN04] Yevgeny Kazakov and Hans de Nivelle. A Resolution Decision Procedure for the Guarded Fragment with Transitive Guards. In David Basin and Michaël Rusinowitch, editors, *Automated Reasoning*. Volume 3097, Lecture Notes in Computer Science, pages 122–136. Springer Berlin Heidelberg, 2004 (cited on page 28).

[KFNM04] Holger Knublauch, Ray W. Fergerson, Natalya F. Noy, and Mark A. Musen. The Protégé OWL Plugin: An Open Development Environment for Semantic Web Applications. In Sheila A. McIlraith, Dimitris Plexousakis, and Frank van Harmelen, editors, *Proceedings of the 3rd International Semantic Web Conference (ISWC 2004)*, volume 3298 of *LNCS*, pages 229–243. Springer, 2004 (cited on page 91).

[KG13] Ilianna Kollia and Birte Glimm. Optimizing SPARQL Query Answering over OWL Ontologies. *Journal of Artificial Intelligence Research*, 48(1):253–303, 2013 (cited on page 113).

[KKS11] Yevgeny Kazakov, Markus Krötzsch, and František Simančík. Concurrent Classification of $\mathcal{EL}$ Ontologies. In Lora Aroyo, Chris Welty, Harith Alani, Jamie Taylor,

Abraham Bernstein, Lalana Kagal, Natasha Noy, and Eva Blomqvist, editors, *Proceedings of the 10th International Semantic Web Conference (ISWC 2011)*, volume 7032 of *LNCS*, pages 305–320. Springer, 2011 (cited on pages 4, 31).

[KKS13] Yevgeny Kazakov, Markus Krötzsch, and František Simančík. The Incredible ELK: From Polynomial Procedures to Efficient Reasoning with $\mathcal{EL}$ Ontologies. *Journal of Automated Reasoning*, 53:1–61, 1, 2013 (cited on pages 28, 30).

[KM08a] Yevgeny Kazakov and Boris Motik. A Resolution-Based Decision Procedure for $\mathcal{SHOIQ}$. *Journal of Automated Reasoning*, 40(2–3):89–116, 2008 (cited on pages 28, 114).

[KM08b] Yevgeny Kazakov and Boris Motik. A Resolution-Based Decision Procedure for $\mathcal{SHOIQ}$. *Journal of Automated Reasoning*, 40(2–3):89–116, 2008 (cited on page 113).

[KSH12] Markus Krötzsch, František Simančík, and Ian Horrocks. A Description Logic Primer. *CoRR*, abs/1201.4089, 2012 (cited on page 15).

[LAFG+05] Lee Lacy, Gabriel Aviles, Karen Fraser, William Gerber, Alice M. Mulvehill, and Robert Gaskill. Experiences Using OWL in Military Applications. In *Proceedings of the 1st International Workshop on OWL Experiences and Directions (OWLED 2005)*, volume 188 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2005 (cited on page 1).

[LB10] Michael J. Lawley and Cyril Bousquet. Fast Classification in Protégé: Snorocket as an OWL 2 EL Reasoner. In Kerry Taylor, Thomas Meyer, and Mehmet Orgun, editors, *Proceedings of the 6th Australasian Ontology Workshop (IAOA 2010)*, volume 122 of *Conferences in Research and Practice in Information Technology*, pages 45–49. Australian Computer Society Inc., 2010 (cited on page 4).

[Leh92] Fritz Lehman, editor. Computers & Mathematics with Applications 23(2–9) (1992) (cited on page 23).

[LST05] Carsten Lutz, Ulrike Sattler, and Lidia Tendera. The complexity of finite model reasoning in Description Logics. *Information and Computation*, 199(1–2):132–171, 2005 (cited on page 19).

[Lut03] Carsten Lutz. Description Logics with Concrete Domains—A Survey. In Philippe Balbiani, Nobu-Yuki Suzuki, Frank Wolter, and Michael Zakharyaschev, editors, *Advances in Modal Logics Volume 4*, pages 265–296. King's College Publications, 2003 (cited on page 112).

[Lut04] Carsten Lutz. NEXPTIME-complete Description Logics with Concrete Domains. *ACM Transactions on Computational Logic*, 5(4):669–705, 2004 (cited on page 112).

[McC10] William McCune. Prover9. `https://www.cs.unm.edu/~mccune/prover9/`, 2005–2010 (cited on pages 92, 96).

[McC94] William McCune. OTTER 3.0 Reference Manual and Guide. Technical report ANL-94/6, Argonne National Laboratory, 1994 (cited on page 92).

[MCHW+09] Boris Motik, Bernardo Cuenca Grau, Ian Horrocks, Zhe Wu, Achille Fokoue, and Carsten Lutz. OWL 2 Web Ontology Language: Profiles, W3C Recommendation. `https://www.w3.org/TR/owl2-profiles/`, 2009 (cited on pages 2, 25).

[Men12] Julian Mendez. jcel: A Modular Rule-based Reasoner. In Ian Horrocks, Mikalai Yatskevich, and Ernesto Jiménez-Ruiz, editors, *Proceedings of the 1st International Workshop on OWL Reasoner Evaluation (ORE 2012)*, volume 858 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2012 (cited on page 4).

[MH08] Boris Motik and Ian Horrocks. Individual Reuse in Description Logic Reasoning. In *Proceedings of the 4th International Joint Conference on Automated Reasoning (IJCAR 2008)*, pages 242–258, 2008 (cited on page 28).

[Mot06] Boris Motik. *Reasoning in Description Logics using Resolution and Deductive Databases*. PhD thesis, Univesität Karlsruhe, Germany, 2006 (cited on pages 60, 96, 112, 114).

[MPP12] Boris Motik, Peter F. Patel-Schneider, and Bijan Parsia. OWL 2 Web Ontology Language: Structural Specification and Functional-Style Syntax (Second Edition), W3C Recommendation. `https://www.w3.org/TR/owl2-syntax/`, 2012 (cited on page 112).

[MSH09] Boris Motik, Rob Shearer, and Ian Horrocks. Hypertableau Reasoning for Description Logics. *Journal of Artificial Intelligence Research*, 36:165–228, 2009 (cited on pages 3, 20, 24, 26–28, 38).

[MW97] William McCune and Larry Wos. Otter: The CADE-13 Competition Incarnations. *Journal of Automated Reasoning*, 18(2):211–220, 1997 (cited on page 92).

[NR01] Robert Nieuwenhuis and Albert Rubio. Paramodulation-Based Theorem Proving. In John Alan Robinson and Andrei Voronkov, editors, *Handbook of Automated Reasoning*. Volume I, chapter 7, pages 371–443. Elsevier Science, 2001 (cited on pages 11, 13, 96, 114).

[NR95] Robert Nieuwenhuis and Albert Rubio. Theorem Proving with Ordering and Equality Constrained Clauses. *Journal of Symbolic Computation*, 19(4):312–351, 1995 (cited on pages 6, 63, 66, 75).

[ORS10] Magdalena Ortiz, Sebastian Rudolph, and Mantas Simkus. Worst-Case Optimal Reasoning for the Horn-DL Fragments of OWL 1 and 2. In Fangzhen Lin, Ulrike Sattler, and Miroslaw Truszczynski, editors, *Proceedings of the 12th International Conference on Principles of Knowledge Representation and Reasoning (KR 2010)*, pages 269–279. AAAI Press, 2010 (cited on pages 4, 30, 39, 60).

[PH02] Jeff Z. Pan and Ian Horrocks. Extending Datatype Support in Web Ontology Reasoning. In *Proceedings of the 1st International Conference on Ontologies, Databases and Applications of SEmantics (ODBASE 2002)*, number 2519 in LNCS, pages 1067–1081. Springer, 2002 (cited on page 112).

[PH12] Laleh Roosta Pour and Volker Haarslev. Algebraic Reasoning for $\mathcal{SHIQ}$. In Yevgeny Kazakov, Domenico Lembo, and Frank Wolter, editors, *Proceedings of the 25th International Workshop on Description Logics (DL 2012)*, volume 846 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2012 (cited on page 114).

[RM03] Cornelius Rosse and José L. V. Mejino Jr. A reference ontology for biomedical informatics: the Foundational Model of Anatomy. *Journal of Biomedical Informatics*, 36(6):478–500, 2003 (cited on pages 1, 24).

[RR06] Alan Rector and Jeremy Rogers. Ontological and Practical Issues in Using a Description Logic to Represent Medical Concept Systems: Experience from GALEN. In Pedro Barahona, François Bry, Enrico Franconi, Nicola Henze, and Ulrike Sattler, editors, *Reasoning Web*. Volume 4126, Lecture Notes in Computer Science, pages 197–231. Springer Berlin Heidelberg, 2006 (cited on pages 1, 24).

[RV02] Alexandre Riazanov and Andrei Voronkov. The design and implementation of VAMPIRE. *AI Communications*, 15(2–3):91–110, 2002 (cited on page 6).

[RV03] Alexandre Riazanov and Andrei Voronkov. Limited resource strategy in resolution theorem proving. *Journal of Symbolic Computation*, 36(1-2):101–115, 2003 (cited on pages 92, 96).

[SARB+07] Barry Smith et al. The OBO Foundry: coordinated evolution of ontologies to support biomedical data integration. *Nature Biotechnology*, 25(11):1251–1255, 2007 (cited on pages 1, 25).

[Sch02] Stephan Schulz. E—A Brainiac Theorem Prover. *AI Communications*, 15(2–3):111–126, 2002 (cited on pages 6, 95).

[Sch13a] Stephan Schulz. *Simple and Efficient Clause Subsumption with Feature Vector Indexing*. In *Automated Reasoning and Mathematics: Essays in Memory of William W. McCune*. Maria Paola Bonacina and Mark E. Stickel, editors. Volume 7788. LNCS. Springer, 2013, pages 45–67 (cited on page 100).

[Sch13b] Stephan Schulz. System Description: E 1.8. In Ken McMillan, Aart Middeldorp, and Andrei Voronkov, editors, *Proceedings of the 19th International Conference on Logic for Programming, Artificial Intelligence, and Reasoning (LPAR-19)*, volume 8312 of *LNCS*, pages 735–743. Springer, 2013 (cited on page 95).

[SCP06] Evren Sirin, Bernardo Cuenca Grau, and Bijan Parsia. From Wine to Water: Optimizing Description Logic Reasoning for Nominals. In *Proceedings of the 10th International Conference on Principles of Knowledge Representation and Reasoning (KR 2006)*, pages 90–99. AAAI Press, 2006 (cited on page 105).

[SDCS05] Amandeep S. Sidhu, Tharam S. Dillon, Elisabeth Chang, and Baldev Singh Sidhu. Protein Ontology Development using OWL. In *Proceedings of the 1st International*

Workshop on OWL Experiences and Directions (OWLED 2005), volume 188 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2005 (cited on page 1).

[SG15] Andreas Steigmiller and Birte Glimm. Pay-As-You-Go Description Logic Reasoning by Coupling Tableau and Saturation Procedures. *Journal of Artificial Intelligence Research*, 54:535–592, 2015 (cited on pages 5, 32).

[SH03] Renate A. Schmidt and Ullrich Hustadt. A Principle for Incorporating Axioms into the First-Order Translation of Modal Formulae. In Franz Baader, editor, *Proceedings of the 19th International Conference on Automated Deduction (CADE-19)*, volume 2741 of *LNAI*, pages 412–426. Springer, 2003 (cited on page 60).

[SKH11] František Simančík, Yevgeny Kazakov, and Ian Horrocks. Consequence-Based Reasoning beyond Horn Ontologies. In Toby Walsh, editor, *Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI 2011)*, pages 1093–1098. AAAI Press/IJCAI, 2011 (cited on pages 4, 30, 31, 39).

[SLG14] Andreas Steigmiller, Thorsten Liebig, and Birte Glimm. Konclude: System Description. *Journal of Web Semantics: Science, Services and Agents on the World Wide Web*, 27:78–85, 2014 (cited on pages 3, 106).

[SLLF+04] Dagobert Soergel, Boris Lauser, Anita Liang, Frehiwot Fisseha, Johannes Keizer, and Stephen Katz. Reengineering Thesauri for New Applications: the AGROVOC Example. *Journal of Digital Information*, 4(4), 2004 (cited on page 1).

[SMH14] František Simančík, Boris Motik, and Ian Horrocks. Consequence-Based and Fixed-Parameter Tractable Reasoning in Description Logics. *Artificial Intelligence*, 209:29–77, 2014 (cited on pages 4, 30, 39, 42, 43, 53).

[Spa00] Kent A. Spackman. Managing clinical terminology hierarchies using algorithmic calculation of subsumption: Experience with SNOMED-RT. *Journal of the American Medical Informatics Association*, Fall Symposium Special Issue, 2000 (cited on pages 1, 2, 24).

[SPCK+07] Evren Sirin, Bijan Parsia, Bernardo Cuenca Grau, Aditya Kalyanpur, and Yarden Katz. Pellet: A Practical OWL-DL Reasoner. *Journal of Web Semantics: Science, Services and Agents on the World Wide Web*, 5(2):51–53, 2007 (cited on pages 3, 24, 27).

[SPS09] Heiner Stuckenschmidt, Christine Parent, and Stefano Spaccapietra. *Modular Ontologies: Concepts, Theories and Techniques for Knowledge Modularization*. Springer, 2009 (cited on page 32).

[SRV01] R. C. Sekar, I. V. Ramakrishnan, and Andrei Voronkov. Term Indexing. In John Alan Robinson and Andrei Voronkov, editors, *Handbook of Automated Reasoning*. Volume II, chapter 26, pages 1853–1964. Elsevier Science, 2001 (cited on page 97).

[SS91] Manfred Schmidt-Schauß and Gert Smolka. Attributive concept descriptions with complements. *Artificial Intelligence*, 48(1):1–26, 1991 (cited on page 24).

[SSD13] Weihong Song, Bruce Spencer, and Weichang Du. Complete Classification of Complex $\mathcal{ALCHO}$ Ontologies Using a Hybrid Reasoning Approach. In Thomas Eiter, Birte Glimm, Yevgeny Kazakov, and Markus Krötzsch, editors, *Informal Proceedings of the 26th International Workshop on Description Logics (DL 2013)*, volume 1014 of *CEUR Workshop Proceedings*, pages 942–961. CEUR-WS.org, 2013 (cited on page 32).

[Sun09] Boontawee Suntisrivaraporn. *Polynomial-Time Reasoning Support for Design and Maintenance of Large-Scale Biomedical Ontologies*. PhD thesis, TU Dresden, Germany, 2009 (cited on page 30).

[TH04] Dmitry Tsarkov and Ian Horrocks. Efficient Reasoning with Range and Domain Constraints. In *Proceedings of the 17th International Workshop on Description Logics (DL 2004)*, pages 41–50, 2004 (cited on page 26).

[TH06] Dmitry Tsarkov and Ian Horrocks. FaCT++ Description Logic Reasoner: System Description. In Ulrich Furbach and Natarajan Shankar, editors, *Proceedings of the 3rd International Joint Conference on Automated Reasoning (IJCAR 2006)*, volume 4130 of *LNAI*, pages 292–297. Springer, 2006 (cited on pages 3, 24, 27).

[THP07] Dmitry Tsarkov, Ian Horrocks, and Peter F. Patel-Schneider. Optimizing Terminological Reasoning for Expressive Description Logics. *Journal of Automated Reasoning*, 39(3):277–316, 2007 (cited on page 26).

[Tob01] Stephan Tobies. *Complexity Results and Practical Algorithms for Logics in Knowledge Representation*. PhD thesis, LuFG Theoretical Computer Science, RWTH Aachen, Germany, 2001 (cited on pages 19, 21, 24, 113).

[TP12] Dmitry Tsarkov and Ignazio Palmisano. Chainsaw: a Metareasoner for Large Ontologies. In Ian Horrocks, Mikalai Yatskevich, and Ernesto Jiménez-Ruiz, editors, *Proceedings of the 1st International Workshop on OWL Reasoner Evaluation (ORE 2012)*, volume 858 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2012 (cited on page 32).

[Var97] Moshe Y. Vardi. Why is Modal Logic So Robustly Decidable? In Neil Immerman and Phokion G. Kolaitis, editors, *Descriptive Complexity and Finite Models*, volume 31 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pages 149–184. American Mathematical Society, 1997 (cited on page 26).

[VDHJ16] Jelena Vlasenko, Maryam Daryalal, Volker Haarslev, and Brigitte Jaumard. A Saturation-based Algebraic Reasoner for $\mathcal{ELQ}$. In Pascal Fontaine, Stephan Schulz, and Josef Urban, editors, *Proceedings of the 5th Workshop on Practical Aspects of Automated Reasoning*, volume 1635 of *CEUR Workshop Proceedings*, pages 110–124. CEUR-WS.org, 2016 (cited on page 114).

[vHLP07] Frank van Harmelen, Vladimir Lifschitz, and Bruce Porter. *Handbook of Knowledge Representation*. Foundations of Artificial Intelligence. Elsevier Science, 2007 (cited on pages 1, 25).