Extending Consequence-Based Reasoning to $\mathcal{SRIQ}$

Andrew Bate, Boris Motik, Bernardo Cuenca Grau, František Simančík, Ian Horrocks
Department of Computer Science, University of Oxford,
Oxford, United Kingdom
firstname.lastname@cs.ox.ac.uk

Abstract

Consequence-based calculi are a family of reasoning algorithms for *description logics* (DLs), and they combine hypertableau and resolution in a way that often achieves excellent performance in practice. Up to now, however, they were proposed for either Horn DLs (which do not support disjunction), or for DLs without counting quantifiers. In this paper we present a novel consequence-based calculus for $\mathcal{SRIQ}$—a rich DL that supports both features. This extension is non-trivial since the intermediate consequences that need to be derived during reasoning cannot be captured using DLs themselves. The results of our preliminary performance evaluation suggest the feasibility of our approach in practice.

1 Introduction

Description logics (DLs) (Baader et al. 2003) are a family of knowledge representation formalisms with numerous applications in practice. DL-based applications model a domain of interest by means of an *ontology*, in which key notions in the domain are described using *concepts* (i.e., unary predicates), and the relationships between concepts are described using *roles* (i.e., binary predicates). *Subsumption* is the problem of determining whether each instance of a concept C is also an instance of a concept D in all models of an ontology, and it is a fundamental reasoning problem in applications of DLs. For expressive DLs, this problem is of high worst-case complexity, ranging from ExpTime up to N2ExpTime.

Despite these discouraging complexity bounds, highly optimised reasoners such as FaCT++ (Tsarkov and Horrocks 2006), Pellet (Sirin et al. 2007), HermiT (Glimm et al. 2014), and Konclude (Steigmiller, Liebig, and Glimm 2014) have proved successful in practice. These systems are typically based on (hyper)tableau calculi, which construct a finite representation of a canonical model of the ontology 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.

A recent breakthrough in DL reasoning came in the form of *consequence-based calculi*. The reasoning algorithm by Baader, Brandt, and Lutz (2005) for the lightweight logic $\mathcal{EL}$ can be seen as the first such calculus. It was later extended to the more expressive DLs Horn-$\mathcal{SHIQ}$ (Kazakov 2009) and Horn-$\mathcal{SROIQ}$ (Ortiz, Rudolph, and Simkus 2010)—DLs that support counting quantifiers, but not disjunctions between concepts. Consequence-based calculi were also developed for $\mathcal{ALCH}$ (Simančík, Kazakov, and Horrocks 2011) and $\mathcal{ALCI}$ (Simančík, Motik, and Horrocks 2014), which support concept disjunction, but not counting quantifiers. Such calculi can be seen as combining resolution and hypertableau (see Section 3 for details): as in resolution, they describe ontology models by systematically deriving relevant consequences; and as in (hyper)tableau, they are goal-directed and avoid drawing unnecessary consequences. Additionally, they are not only refutationally complete, but can also (dis)prove all relevant subsumptions in a single run, which can greatly reduce the overall computational work. Finally, unlike implemented (hyper)tableau reasoners, they are worst-case optimal for the logic they support. Steigmiller, Glimm, and Liebig (2014) presented a way of combining a consequence-based calculus with a traditional tableau-based prover; while such a combination seems to perform well in practice, the saturation rules are only known to be complete for $\mathcal{EL}$ ontologies, and the overall approach is not worst-case optimal for $\mathcal{SRIQ}$.

Existing consequence-based algorithms cannot handle DLs such as $\mathcal{ALCHIQ}$ that provide both disjunctions and counting quantifiers. As we argue in Section 3, extending these algorithms to handle such DLs is challenging: counting quantifiers require equality reasoning which, together with disjunctions, can impose complex constraints on ontology models; and, unlike existing consequence-based calculi, such constraints cannot be captured using DLs themselves, which makes the reasoning process much more involved.

In Section 4 we present a consequence-based calculus for $\mathcal{ALCHIQ}$; by using the encoding of role chains by Kazakov (2008), our calculus can also handle $\mathcal{SRIQ}$, which covers all of OWL 2 DL except for nominals, reflexive roles, and datatypes. Borrowing ideas from resolution theorem proving, we encode the calculus' consequences as first-order clauses of a specific form, and we handle equality using a variant of *ordered paramodulation* (Nieuwenhuis and Rubio 1995)—a state of the art calculus for equational theorem proving used in modern theorem provers such as E (Schulz 2002) and Vampire (Riazanov and Voronkov 2002). 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.

We have implemented a prototype system and compared its performance with that of well-established reasoners. Our results in Section 5 suggest that our system can significantly outperform FaCT++, Pellet, or HermiT, and often exhibits comparable performance to that of Konclude.

2 Preliminaries

First-Order Logic. It is usual in equational theorem proving to encode atomic formulas as terms, and to use a multi-sorted signature that prevents us from considering malformed terms. Thus, we partition the signature into a set $\mathcal{P}$ of *predicate symbols* and a set $\mathcal{F}$ of *function symbols*; moreover, we assume that $\mathcal{P}$ has a special constant $\wp$. A *term* is constructed as usual using variables and the signature symbols, with the restriction that predicate symbols are allowed to occur only at the outermost level; the latter terms are called $\mathcal{P}$*-terms*, while all other terms are $\mathcal{F}$*-terms*. For example, for P a predicate and f a function symbol, $f(P(x))$ and $P(P(x))$ are both malformed; $P(f(x))$ is a well-formed $\mathcal{P}$-term; and $f(x)$ and x are both well-formed $\mathcal{F}$-terms. Term $f(t)$ is an f*-successor* of t, and t is an f*-predecessor* of $f(t)$.

An *equality* is a formula of the form $s \approx t$, where s and t are either both $\mathcal{F}$- or both $\mathcal{P}$-terms. An equality of the form $P(\vec{s}) \approx \wp$ is called an *atom* and is written as just $P(\vec{s})$ whenever it is clear from the context that the expression denotes a formula, and not a $\mathcal{P}$-term. An *inequality* is a negation of an equality and is written as $s \not\approx t$. We assume that $\approx$ and $\not\approx$ are implicitly symmetric—that is, $s \bowtie t$ and $t \bowtie s$ are identical, for $\bowtie \in \{\approx, \not\approx\}$. A *literal* is an equality or an inequality. A *clause* is a formula of the form $\forall \vec{x}.[\Gamma \rightarrow \Delta]$ where Γ is a conjunction of atoms 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 omitted as it is understood implicitly. We often treat conjunctions and disjunctions as sets (i.e., they are unordered and without repetition) and use them in standard set operations; and we write the empty conjunction (disjunction) as $\top$ ($\bot$). For α a term, literal, clause, or a set thereof, we say that α is *ground* if it does not contain a variable; $\alpha\sigma$ is the result of applying a substitution σ to α; and we often write substitutions as $\sigma = \{x \mapsto t_1,\ y \mapsto t_2,\ \ldots\}$. We use the standard notion of subterm positions; $s|_p$ is the subterm of s at position p; position p is *proper* in a term t if $t|_p \neq t$; and $s[t]_p$ is the term obtained by replacing the subterm of s at position p with t.

A *Herbrand equality interpretation* is a set of ground equalities satisfying the usual congruence properties. Satisfaction of a ground conjunction, a ground disjunction, or a (not necessarily ground) clause α in an interpretation I, written $I \models \alpha$, as well as entailment of a clause $\Gamma \rightarrow \Delta$ from a set of clauses $\mathcal{O}$, written $\mathcal{O} \models \Gamma \rightarrow \Delta$, are defined as usual. Note that a ground disjunction of literals Δ may contain inequalities so $I \models \Delta$ does not necessarily imply $I \cap \Delta \neq \emptyset$.

Unless otherwise stated, (possibly indexed) letters x, y, and z denote variables; l, r, s, and t denote terms; A denotes an atom or a $\mathcal{P}$-term (depending on the context); L denotes a literal; f and g denote function symbols; B denotes a unary predicate symbol; and S denotes a binary predicate symbol.

Orders. A *strict order* $\succ$ on a universe U is an irreflexive, asymmetric, and transitive relation on U; and $\succeq$ is the *non-strict order* induced by $\succ$. Order $\succ$ is *total* if, for all $a, b \in U$, we have $a \succ b$, $b \succ a$, or $a = b$. Given $\circ \in \{\succ, \succeq\}$, element $b \in U$, and subset $S \subseteq U$, the notation $S \circ b$ abbreviates $\exists a \in S : a \circ b$. The *multiset extension* $\succ_{mul}$ of $\succ$ compares multisets M and N on U such that $M \succ_{mul} N$ 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 set of all terms. 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.

DL-Clauses. Our calculus takes as input a set $\mathcal{O}$ of *DL-clauses*—that is, clauses restricted to the following form. Let $\mathcal{P}_1$ and $\mathcal{P}_2$ be countable sets of unary and binary predicate symbols, and let $\mathcal{F}$ be a countable set of unary function symbols. DL-clauses are written using the *central variable* x and variables z_i. A *DL-*$\mathcal{F}$*-term* has the form x, z_i, or $f(x)$ with $f \in \mathcal{F}$; a *DL-*$\mathcal{P}$*-term* has the form $B(z_i)$, $B(x)$, $B(f(x))$, $S(x, z_i)$, $S(z_i, x)$, $S(x, f(x))$, $S(f(x), x)$ with $B \in \mathcal{P}_1$ and $S \in \mathcal{P}_2$; and a *DL-term* is a DL-$\mathcal{F}$-term or a DL-$\mathcal{P}$-term. A *DL-atom* has the form $A \approx \wp$ with A a DL-$\mathcal{P}$-term. A *DL-literal* is a DL-atom, or it is of the form $f(x) \bowtie g(x)$, $f(x) \bowtie z_i$, or $z_i \bowtie z_j$ with $\bowtie \in \{\approx, \not\approx\}$. A *DL-clause* contains only DL-atoms of the form $B(x)$, $S(x, z_i)$, and $S(z_i, x)$ in the body and only DL-literals in the head, and each variable z_i occurring in the head 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 $B(x)$. Given an ontology $\mathcal{O}$ and a query clause $\Gamma \rightarrow \Delta$, our calculus decides whether $\mathcal{O} \models \Gamma \rightarrow \Delta$ holds.

$\mathcal{SRIQ}$ ontologies written using the DL-style syntax can be transformed into DL-clauses without affecting query clause entailment. First, we *normalise* DL axioms to the form shown on the left-hand side of Table 1: we transform away role chains and then replace all complex concepts with fresh atomic ones; this process is well understood (Kazakov 2009; 2008; Simančík, Motik, and Horrocks 2014), so we omit the details. Second, using the well-known correspondence between DLs and first-order logic (Baader et al. 2003), we translate normalised axioms to DL-clauses as shown on the right-hand side of Table 1. The standard translation of $B_1 \sqsubseteq \leqslant n\, S.B_2$ requires atoms $B_2(z_i)$ in clause bodies, which are not allowed in our setting. We address this issue by introducing a fresh role S_{B_2} that we axiomatise as $S(y, x) \wedge B_2(x) \rightarrow S_{B_2}(y, x)$; this, in turn, allows us to clausify the original axiom as if it were $B_1 \sqsubseteq \leqslant n\, S_{B_2}$. For an $\mathcal{ELH}$ ontology, $\mathcal{O}$ contains DL-clauses of type DL1 with $m = n + 1$, DL2 with $n = 1$, DL3, and DL5.

3 Motivation

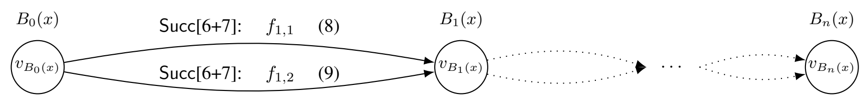
As motivation for our work, in Section 3.1 we discuss the drawbacks of existing DL reasoning calculi, and then in Section 3.2 we discuss how existing consequence-based calculi

Table 1: Translating Normalised $\mathcal{ALCHIQ}$ Ontologies into DL-Clauses

DL1	$\bigsqcap_{1\leq i\leq n} B_i \sqsubseteq \bigsqcup_{n+1\leq i\leq m} B_i$	$\rightsquigarrow$	$\bigwedge_{1\leq i\leq n} B_i(x) \rightarrow \bigvee_{n+1\leq i\leq m} B_i(x)$	
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	$S_1 \sqsubseteq S_2^-$	$\rightsquigarrow$	$S_1(z_1, x) \rightarrow S_2(x, z_1)$	

Ontology $\mathcal{O}_1$

$B_i \sqsubseteq \exists S_j.B_{i+1}$	$\rightsquigarrow$	$B_i(x) \rightarrow S_j(x, f_{i+1,j}(x))$	(1)	for $0 \leq i < n$ and $1 \leq j \leq 2$
		$B_i(x) \rightarrow B_{i+1}(f_{i+1,j}(x))$	(2)	
$B_n \sqsubseteq C_n$	$\rightsquigarrow$	$B_n(x) \rightarrow C_n(x)$	(3)	
$\exists S_j.C_{i+1} \sqsubseteq C_i$	$\rightsquigarrow$	$S_j(z_1, x) \wedge C_{i+1}(x) \rightarrow C_i(z_1)$	(4)	for $0 \leq i < n$ and $1 \leq j \leq 2$



Initialisation:	$\top \rightarrow B_0(x)$	(5)
Hyper[1+5]:	$\top \rightarrow S_j(x, f_{1,j}(x))$	(6)
Hyper[2+5]:	$\top \rightarrow B_1(f_{1,j}(x))$	(7)
Pred[19]:	$\top \rightarrow C_0(x)$	(20)

Succ[6+7]:	$\top \rightarrow S_j(y, x)$	(10)
Succ[6+7]:	$\top \rightarrow B_1(x)$	(11)
Hyper[1+11]:	$\top \rightarrow S_j(x, f_{2,j}(x))$	(12)
Hyper[2+11]:	$\top \rightarrow B_2(x, f_{2,j}(x))$	(13)
Pred[...]:	$\top \rightarrow C_1(x)$	(18)
Hyper[4+10+18]:	$\top \rightarrow C_0(y)$	(19)

Succ[...]:	$\top \rightarrow S_j(y, x)$	(14)
Succ[...]:	$\top \rightarrow B_n(x)$	(15)
Hyper[3+15]:	$\top \rightarrow C_n(x)$	(16)
Hyper[4+14+16]:	$\top \rightarrow C_{n-1}(y)$	(17)

Figure 1: Example Motivating Consequence-Based Calculi

address these problems by separating clauses into contexts in a way that considerably reduces the number of inferences. Next, in Section 3.3 we discuss the main contribution of this paper, which lies in extending the consequence-based framework to a DL with disjunctions and number restrictions. Handling the latter requires equality reasoning, which requires a more involved calculus and completeness proof.

3.1 Why Consequence-Based Calculi?

Consider the $\mathcal{EL}$ ontology $\mathcal{O}_1$ in Figure 1; one can readily check that $\mathcal{O} \models B_i(x) \rightarrow C_i(x)$ holds for $0 \leq i \leq n$. To prove $\mathcal{O} \models B_0(x) \rightarrow C_0(x)$ using the (hyper)tableau calculus, we start with $B_0(a)$ and apply (1)–(4) in a forward-chaining manner. Since $\mathcal{O}$ contains (1) for $j \in \{1, 2\}$, this 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 labelled with B_i are of the same type and they share the same properties, 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. Techniques such as *caching* (Goré and Nguyen 2007) or *anywhere blocking* (Motik, Shearer, and Horrocks 2009) can constrain model construction, but their effectiveness often depends on the order of rule applications. Thus, model size is a key limiting factor for (hyper)tableau-based reasoners (Motik, Shearer, and Horrocks 2009).

In contrast, resolution describes models using (universally quantified) clauses that 'summarise' the model. This eliminates redundancy and ensures worst-case optimality of many resolution decision procedures. Many resolution variants have been proposed (Bachmair and Ganzinger 2001), each restricting inferences in a specific way. However, to ensure termination, all decision procedure for DLs we are aware of perform inferences with the 'deepest' and the 'covering' clause atoms, so all of them will resolve all (1) with all (4) to obtain all $2n^2$ clauses of the form

$$B_i(x) \wedge C_{k+1}(f_{i+1,j}(x)) \rightarrow C_k(x) \quad \text{for } 1 \leq i, k < n \text{ and } 1 \leq j \leq 2. \tag{21}$$

Of these $2n^2$ clauses, only those with $i = k$ are relevant to proving our goal. If we extend $\mathcal{O}$ 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 $\mathcal{O}$ is satisfiable since we must then produce all consequences of $\mathcal{O}$.

3.2 Basic Notions

Consequence-based calculi combine 'summarisation' of resolution with goal-directed search of (hyper)tableau calculi. Simančík, Motik, and Horrocks (2014) presented a framework for $\mathcal{ALCI}$ capturing the key elements of the related calculi by Baader, Brandt, and Lutz (2005), Kazakov (2009), Ortiz, Rudolph, and Simkus (2010), and Simančík, Kazakov, and Horrocks (2011). Before extending this framework to $\mathcal{ALCHIQ}$ in Section 4, we next informally recapitulate the basic notions; however, to make this paper easier to follow, we use the same notation and terminology as in Section 4.

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*. The vertices in $\mathcal{V}$ are called *contexts*. Let I be a Herbrand model of $\mathcal{O}$; hence, the domain of I contains ground terms. Instead of representing each ground term of I separately as in (hyper)tableau calculi, $\mathcal{D}$ can represent the properties of several terms by a single context v. Each context $v \in \mathcal{V}$ is associated with a (possibly empty) conjunction core_v of *core* atoms that must hold for all ground terms that v represents; thus, core_v determines the 'kind' of context v. Moreover, v is associated with a set $\mathcal{S}_v$ of clauses that capture the constraints that these terms must satisfy. Partitioning clauses into sets allows us to restrict the inferences between clause sets and thus eliminate certain irrelevant inferences. Clauses in $\mathcal{S}_v$ are 'relative' to core_v: for each $\Gamma \to \Delta \in \mathcal{S}_v$, we have $\mathcal{O} \models \mathsf{core}_v \wedge \Gamma \to \Delta$—that is, we do not include core_v in clause bodies since core_v holds implicitly. Function $\succ$ provides each context $v \in \mathcal{V}$ with a concept order $\succ_v$ that restricts resolution inferences in the presence of disjunctions.

Contexts are connected by directed edges labelled with function symbols. If u is connected to v via an f-labelled edge, then the f-successor of each ground term represented by u is represented by v. 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 derive the required consequences. Figure 1 demonstrates this for $\mathcal{O}_1 \models B_0(x) \to C_0(x)$. The cores and the clauses shown above and below, respectively, each context, and clause numbers correspond to the derivation order. To prove $B_0(x) \to C_0(x)$, we introduce context $v_{B_0(x)}$ with core $B_0(x)$ and add clause (5) to it. The latter says that B_0 holds for a, and it is analogous to initialising a (hyper)tableau calculus with $B_0(a)$. The calculus then applies rules from Table 2 to derive new clauses and/or extend $\mathcal{D}$.

Hyper is the standard hyperresolution rule restricted to a single context at a time. Thus, we derive (6) from (1) and (5), and (7) from (2) and (5). Hyperresolution resolves all body atoms, which makes the resolvent relevant for the context and prevents the derivation of irrelevant clauses such as (21).

Context $v_{B_0(x)}$ contains atoms with function symbols $f_{1,1}$ and $f_{1,2}$, so the Succ rule must ensure that the $f_{1,1}$- and $f_{1,2}$-successors of the ground terms represented by $v_{B_0(x)}$ are adequately represented in $\mathcal{D}$. We can control context introduction via a parameter called an *expansion strategy*—a function that determines whether to reuse an existing context or introduce a fresh one; in the latter case, it also determines how to initialise the context's core. We discuss possible strategies in Section 4.1; in the rest of this example, we use the so-called cautious strategy, where the Succ rule introduces context $v_{B_1(x)}$ and initialises it with (10) and (11). Note that (6) represents two clauses, both of which we satisfy (in separate applications of the Succ rule) using $v_{B_1(x)}$.

We construct contexts $v_{B_2(x)}, \ldots, v_{B_n(x)}$ analogously, we derive (16) by hyperresolving (3) and (14), and we derive (17) by hyperresolving (4), (14), and (16). Clause (17) imposes a constraint on the predecessor context, which we propagate using the Pred rule, deriving (19) and (20). Since clauses of $v_{B_0(x)}$ are 'relative' to the core of $v_{B_0(x)}$, clause (20) represents our query clause, as required.

3.3 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, Motik, and Horrocks (2014) represented all relevant consequences using DL axioms of the following form:

$$\bigsqcap B_i \sqsubseteq \bigsqcup B_j \sqcup \bigsqcup \exists S_k.B_k \sqcup \bigsqcup \forall S_\ell.B_\ell \qquad (55)$$

$\mathcal{ALCHIQ}$ provides both counting quantifiers and disjunctions, the interplay of which may impose constraints that cannot be represented in $\mathcal{ALCHIQ}$. Let $\mathcal{O}_2$ be as in Figure 2. To see that $\mathcal{O}_2 \models B_0(x) \to B_4(x)$ holds, we construct a Herbrand interpretation I from $B_0(a)$: (22) and (23) derive $S(f_1(a), a)$ and $B_1(f_1(a))$; and (25) and (26) derive $S(f_1(a), f_2(f_1(a)))$ and $B_2(f_2(f_1(a)))$, and $S(f_1(a), f_3(f_1(a)))$ and $B_3(f_3(f_1(a)))$. Due to (27) we derive $B_4(f_2(f_1(a)))$ and $B_4(f_3(f_1(a)))$. Finally, from (28) we derive the following clause:

$$f_2(f_1(a)) \approx a \vee f_3(f_1(a)) \approx a \vee f_3(f_1(a)) \approx f_2(f_1(a)) \qquad (56)$$

Disjunct $f_3(f_1(a)) \approx f_2(f_1(a))$ cannot be satisfied due to (24); 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 (56) and its consequences, but standard DL axioms cannot explicitly refer to specific successors and predecessors. Instead, we capture consequences using *context clauses*—clauses over terms x, $f_i(x)$, and y, where variable x represents the ground terms that a context stands for, $f_i(x)$ represents f_i-successors of x, and y represents the predecessor of x. We can thus identify the predecessor and the successors of x 'by name', allowing us to capture constraint (56) as

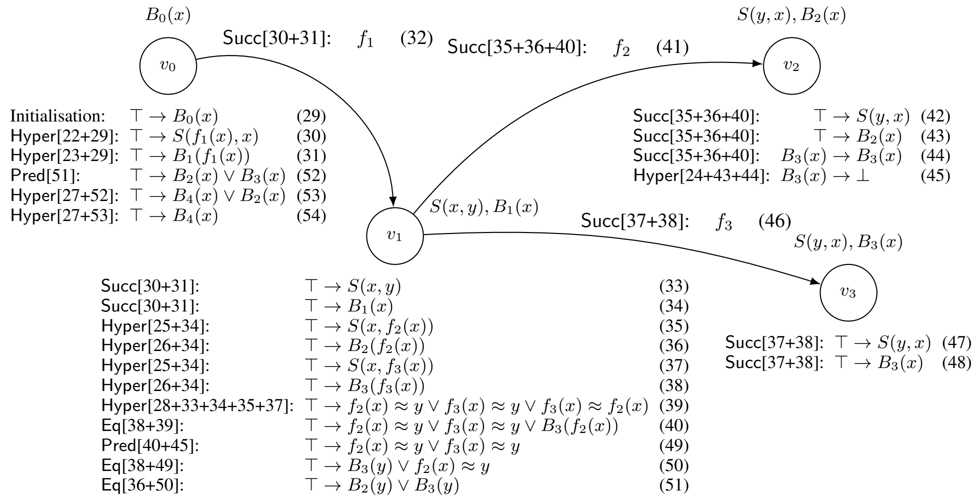
$$f_2(x) \approx y \vee f_3(x) \approx y \vee f_3(x) \approx f_2(x). \qquad (57)$$

Based on this idea, we adapted the rules by Simančík, Motik, and Horrocks (2014) to handle context clauses correctly, and we added rules that capture the consequences of equality. The resulting set of rules is shown in Table 2.

Figure 2 shows how to verify $\mathcal{O}_2 \models B_0(x) \to B_4(x)$ using our calculus; the maximal literal of each clause is shown on the right. We next discuss the inferences in detail.

Ontology $\mathcal{O}_2$

$B_0 \sqsubseteq \exists S^-.B_1$	$\rightsquigarrow$	$B_0(x) \to S(f_1(x), x)$	(22)	
		$B_0(x) \to B_1(f_1(x))$	(23)	
$B_1 \sqsubseteq \exists S.B_i$	$\rightsquigarrow$	$B_1(x) \to S(x, f_i(x))$	(25)	for $2 \leq i \leq 3$
		$B_1(x) \to B_i(f_i(x))$	(26)	for $2 \leq i \leq 3$
$B_i \sqsubseteq B_4$	$\rightsquigarrow$	$B_i(x) \to B_4(x)$	(27)	for $2 \leq i \leq 3$
$B_2 \sqcap B_3 \sqsubseteq \bot$	$\rightsquigarrow$	$B_2(x) \wedge B_3(x) \to \bot$	(24)	
$B_1 \sqsubseteq \leq 2.S$	$\rightsquigarrow$	$B_1(x) \wedge \bigwedge_{1 \leq i \leq 3} S(x, z_i) \to \bigvee_{1 \leq j < k \leq 3} z_j \approx z_k$	(28)	



Initialisation:	$\top \to B_0(x)$	(29)
Hyper[22+29]:	$\top \to S(f_1(x), x)$	(30)
Hyper[23+29]:	$\top \to B_1(f_1(x))$	(31)
Pred[51]:	$\top \to B_2(x) \vee B_3(x)$	(52)
Hyper[27+52]:	$\top \to B_4(x) \vee B_2(x)$	(53)
Hyper[27+53]:	$\top \to B_4(x)$	(54)

Succ[35+36+40]:	$\top \to S(y, x)$	(42)
Succ[35+36+40]:	$\top \to B_2(x)$	(43)
Succ[35+36+40]:	$B_3(x) \to B_3(x)$	(44)
Hyper[24+43+44]:	$B_3(x) \to \bot$	(45)

Succ[30+31]:	$\top \to S(x, y)$	(33)
Succ[30+31]:	$\top \to B_1(x)$	(34)
Hyper[25+34]:	$\top \to S(x, f_2(x))$	(35)
Hyper[26+34]:	$\top \to B_2(f_2(x))$	(36)
Hyper[25+34]:	$\top \to S(x, f_3(x))$	(37)
Hyper[26+34]:	$\top \to B_3(f_3(x))$	(38)
Hyper[28+33+34+35+37]:	$\top \to f_2(x) \approx y \vee f_3(x) \approx y \vee f_3(x) \approx f_2(x)$	(39)
Eq[38+39]:	$\top \to f_2(x) \approx y \vee f_3(x) \approx y \vee B_3(f_2(x))$	(40)
Pred[40+45]:	$\top \to f_2(x) \approx y \vee f_3(x) \approx y$	(49)
Eq[38+49]:	$\top \to B_3(y) \vee f_2(x) \approx y$	(50)
Eq[36+50]:	$\top \to B_2(y) \vee B_3(y)$	(51)

Succ[37+38]:	$\top \to S(y, x)$	(47)
Succ[37+38]:	$\top \to B_3(x)$	(48)

Figure 2: Challenges in Extending the Consequence-Based Framework to $\mathcal{ALCHIQ}$

We first create context v_0 and initialise it with (29); this ensures that each interpretation represented by the context structure contains a ground term for which B_0 holds. Next, we derive (30) and (31) using hyperresolution. At this point, we could hyperresolve (25) and (31) to obtain $\top \to S(f_1(x), f_2(f_1(x)))$; however, this could easily lead to nontermination 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; thus, hyperresolution derives in each context only consequences about x, which prevents redundant derivations.

The Succ rule next handles function symbol f_1 in clauses (30) and (31). To determine which information to propagate to a successor, Definition 2 in Section 4 introduces a set $\mathsf{Su}(\mathcal{O})$ of *successor triggers*. In our example, DL-clause (28) 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 (28) will be interested in information about its predecessors, which we reflect by adding $B_1(x)$ and $S(x, y)$ to $\mathsf{Su}(\mathcal{O})$. In this example we use the so-called eager strategy (see Section 4.1), so the Succ rule introduces context v_1, sets its core to $B_1(x)$ and $S(x, y)$, and initialises the context with (33) and (34).

We next introduce (35)–(38) using hyperresolution, at which point we have sufficient information to apply hyperresolution to (28) to derive (39). Please note how the presence of (33) is crucial for this inference.

We use paramodulation to deal with equality in clause (39). 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 (40).

Clauses (35), (36), and (40) contain function symbol f_2, so the Succ rule introduces context v_2. Due to clause (36), $B_2(x)$ holds for all ground terms that v_2 represents; thus, we add $B_2(x)$ to core_{v_2}. In contrast, atom $B_3(f_2(x))$ occurs in clause (40) in a disjunction, which means it may not hold in v_2; hence, we add $B_3(x)$ to the body of clause (44). The latter clause allows us to derive (45) using hyperresolution.

Clause (45) essentially says '$B_3(f_2(x))$ should not hold in the predecessor', which the Pred rule propagates to v_1 as clause (49); one can understand this inference as hyperresolution of (40) and (45) 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 (51), which essentially says 'the predecessor must satisfy $B_2(x)$ or $B_3(x)$'. The set $\mathsf{Pr}(\mathcal{O})$ of *predecessor triggers* from Definition 2 identifies this as relevant to v_0: the DL-clauses in (27) contain $B_2(x)$ and $B_3(x)$ in their bodies, which are represented in v_1 as $B_2(y)$ and $B_3(y)$. Hence $\mathsf{Pr}(\mathcal{O})$ contains $B_2(y)$ and $B_3(y)$, allowing the Pred rule to derive (52).

After two more steps, we finally derive our target clause (54). We could not do this if $B_4(x)$ were maximal in (53); thus, we require all atoms in the head of a goal clause to be smallest. A similar observation applies to $\mathsf{Pr}(\mathcal{O})$: if $B_3(y)$ were maximal in (50), we would not derive (51) and propagate it to v_0; thus, all atoms in $\mathsf{Pr}(\mathcal{O})$ must be smallest too.

4 Formalising the Algorithm

In this section, we first present our consequence-based algorithm for $\mathcal{ALCHIQ}$ formally, and then we present an outline of the completeness proof; full proofs are given in Bate et al. (2016).

4.1 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 a Herbrand model, and y represents the predecessor of x; this naming convention is important for the 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.

Definition 1. *A* context $\mathcal{F}$-term *is a term of the form x, y, or $f(x)$ for $f \in \mathcal{F}$; a* context $\mathcal{P}$-term *is a term of the form $B(y)$, $B(x)$, $B(f(x))$, $S(x, y)$, $S(y, x)$, $S(x, f(x))$, or $S(f(x), x)$ for $B, R \in \mathcal{P}$ and $f \in \mathcal{F}$; and a* context term *is an $\mathcal{F}$-term or a $\mathcal{P}$-term. A* context literal *is a literal of the form $A \approx \wp$ (called a* context atom*), $f(x) \bowtie g(x)$, or $f(x) \bowtie y$, $y \bowtie y$, for A a context $\mathcal{P}$-term and $\bowtie \in \{\approx, \not\approx\}$. A* context clause *is a clause with only function-free context atoms in the body, and only context literals in the head.*

Definition 2 introduces 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 context's successor, and it guides the Succ rule whereas $\mathsf{Pr}(\mathcal{O})$ contains atoms that are of interest to a context's predecessor and it guides the Pred rule.

Definition 2. *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*

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

As in resolution, we restrict the inferences using a term order $\succ$. Definition 3 specifies the conditions that the order must satisfy. Conditions 1 and 2 ensure that $\mathcal{F}$-terms are compared uniformly across contexts; however, $\mathcal{P}$-terms can be compared in different ways in different contexts. Conditions 1 through 4 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* (Baader and Nipkow 1998)—a kind of term order commonly used in equational theorem proving. Finally, condition 5 ensures that atoms that might be propagated to a context's predecessor via the Pred rule are smallest, which is important for completeness.

Definition 3. *Let $\gtrdot$ be a total, well-founded order on function symbols. A* context term order $\succ$ *is an order on context terms satisfying the following conditions:*

1. *for each $f \in \mathcal{F}$, we have $f(x) \succ x \succ y$;*
2. *for all $f, g \in \mathcal{F}$ with $f \gtrdot g$, we have $f(x) \succ g(x)$;*
3. *for all terms s_1, s_2, and t and each position p in t, if $s_1 \succ s_2$, then $t[s_1]_p \succ t[s_2]_p$;*
4. *for each term s and each proper position p in s, we have $s \succ s|_p$; and*
5. *for each atom $A \approx \wp \in \mathsf{Pr}(\mathcal{O})$ and each context term $s \notin \{x, y\}$, we have $A \not\succ s$.*

Each term order is extended to a literal order, also written $\succ$, as described in Section 2.

A lexicographic path order (LPO) (Baader and Nipkow 1998) over context $\mathcal{F}$-terms and context $\mathcal{P}$-terms, in which x and y are treated as constants such that $x \succ y$, satisfies conditions 1 through 4. Furthermore, $\mathsf{Pr}(\mathcal{O})$ contains only atoms of the form $B(y)$, $S(x, y)$, and $S(y, x)$, which we can always make smallest in the ordering; thus, condition 5 does not contradict the other conditions. Hence, an LPO that is relaxed for condition 5 satisfies Definition 3, and thus, for any given $\gtrdot$, at least one context term order exists.

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

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

1. *$\{s \approx s',\ s \not\approx s'\} \subseteq \Delta$ or $s \approx s \in \Delta$ for some terms s and s', or*
2. *$\Gamma' \subseteq \Gamma$ 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 2 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 uses this to support clause subsumption.

Proposition 1. *For U a set of clauses and C and C' clauses with $C \,\hat{\in}\, U \setminus \{C\}$ and $C' \,\hat{\in}\, U$, we have $C' \,\hat{\in}\, U \setminus \{C\}$.*

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 ontology in question.

Definition 5. *A* context structure *for an ontology $\mathcal{O}$ is a tuple $\mathcal{D} = \langle \mathcal{V}, \mathcal{E}, \mathcal{S}, \mathsf{core}, \succ \rangle$, where $\mathcal{V}$ is a finite set of* contexts, *$\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V} \times \mathcal{F}$ is a finite set of edges each labelled with a function symbol, function* core *assigns to each context v a conjunction core_v of atoms over the $\mathcal{P}$-terms from*

$\mathsf{Su}(\mathcal{O})$*, function* $\mathcal{S}$ *assigns to each context* v *a finite set* $\mathcal{S}_v$ *of context clauses, and function* $\succ$ *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* $\Gamma \rightarrow \Delta \in \mathcal{S}_v$*, we have* $\mathcal{O} \models \mathsf{core}_v \wedge \Gamma \rightarrow \Delta$.

S2. For each edge $\langle u, v, f\rangle \in \mathcal{E}$*, we have*

$$\mathcal{O} \models \mathsf{core}_u \rightarrow \mathsf{core}_v\{x \mapsto f(x), y \mapsto x\}.$$

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

Definition 6. *An* expansion strategy *is a function* 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 computable in polynomial time and it is a triple* $\langle v, \mathsf{core}', \succ'\rangle$ *where* core' *is a subset of* K*; either* $v \notin \mathcal{V}$ *is a fresh context, or* $v \in \mathcal{V}$ *is an existing context in* $\mathcal{D}$ *such that* $\mathsf{core}_v = \mathsf{core}'$*; and* $\succ'$ *is a context term order.*

Simančík, Motik, and Horrocks (2014) presented two basic strategies, which we can adapt to our setting as follows.

- 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 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 f occurs in $\mathcal{O}$ in exactly one atom of the form $B(f(x))$ and if $B(x) \in K_1$, 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. Context $v_{B(x)}$ is then less constrained, but the number of contexts is at most linear.

Simančík, Motik, and Horrocks (2014) discuss extensively the differences between and the relative merits of the two strategies; although their discussion deals with $\mathcal{ALCI}$ only, their conclusions apply to $\mathcal{SRIQ}$ as well.

We are now ready to show soundness and completeness.

Theorem 1 (Soundness)**.** *For any expansion strategy, applying an inference rule from Table 2 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 2 is applicable to* $\mathcal{O}$ *and* $\mathcal{D}$*. Then,* $\Gamma_Q \rightarrow \Delta_Q \mathrel{\hat{\in}} \mathcal{S}_q$ *holds for each query clause* $\Gamma_Q \rightarrow \Delta_Q$ *and each context* $q \in \mathcal{V}$ *that satisfy conditions C1–C3.*

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

C2. For each atom $A \approx \wp \in \Delta_Q$ *and each context term* $s \notin \{x, y\}$*, if* $A \succ_q s$*, then* $s \approx \wp \in \Delta_Q \cup \mathsf{Pr}(\mathcal{O})$.

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

Conditions C2 and C3 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 $\mathcal{O} \models \Gamma_Q \rightarrow \Delta_Q$.

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

Table 2: Rules of the Consequence-Based Calculus

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

A2. Introduce a context q into $\mathcal{D}$; set $\mathsf{core}_q = \Gamma_Q$; for each $A \in \Gamma_Q$, add $\top \rightarrow A$ 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 2 to $\mathcal{D}$ and $\mathcal{O}$.

A4. $\Gamma_Q \rightarrow \Delta_Q$ holds if and only if $\Gamma_Q \rightarrow \Delta_Q \hat{\in} \mathcal{S}_v$.

Propositions 2 and 3 show that our calculus is worst-case optimal for both $\mathcal{ALCHIQ}$ and $\mathcal{ELH}$.

Proposition 2. *For each expansion strategy that introduces at most exponentially many contexts, algorithm A1–A4 runs in worst-case 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, Brandt, and Lutz (2005).*

4.2 An Outline of the Completeness Proof

To prove Theorem 2, we fix an ontology $\mathcal{O}$, a context structure $\mathcal{D}$, a query clause $\Gamma_Q \rightarrow \Delta_Q$, and a context q such that properties C2 and C3 of Theorem 2 are satisfied and $\Gamma_Q \rightarrow \Delta_Q \hat{\notin} \mathcal{S}_q$ holds, and we construct a Herbrand interpretation that satisfies $\mathcal{O}$ but refutes $\Gamma_Q \rightarrow \Delta_Q$. We reuse techniques from equational theorem proving (Nieuwenhuis and Rubio 1995) 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 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 a 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 $\mathcal{F}$- and $\mathcal{P}$-terms constructed using the symbols in $\mathcal{F}$ and $\mathcal{P}$, and a special constant c; for convenience, let $\mathcal{T}$ be the set of all $\mathcal{F}$-terms from this universe.

We obtain R by unfolding the context structure $\mathcal{D}$ starting from context q: we map each $\mathcal{F}$-term $t \in \mathcal{T}$ 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 $\Gamma_t \rightarrow \Delta_t$: 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 $\Gamma_t \rightarrow \Delta_t$. 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 one 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 properties L1–L3 hold:

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

L2. If $t = c$, then $\Delta_t = \Delta_Q$; and if $t \neq c$, then $\Delta_t \subseteq \mathsf{Pr}_t$.

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

The construction produces a rewrite system R_t such that

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

F2. $R_t^* \not\models \Gamma_t \rightarrow \Delta_t$—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 'inherited' constraints.

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 $C^i = \Gamma^i \rightarrow \Delta^i \vee L^i$, $1 \leq i \leq n$, that is compatible with the context ordering $\succ_v$ 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 5 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 property 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 term $t = f(t')$ for each $f \in \mathcal{F}$.

- 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 v satisfies all preconditions of the rule, so we define $X_t = v$. Moreover, 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 $\Gamma \rightarrow \Delta \in \mathcal{O}$ and a substitution τ that makes the clause ground. W.l.o.g. we can

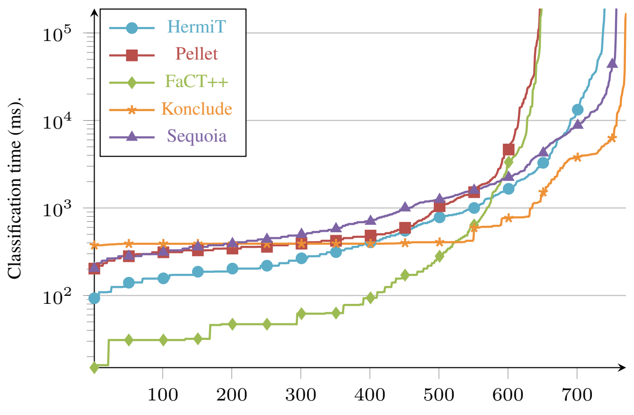


Figure 3: Classification Times for All Ontologies

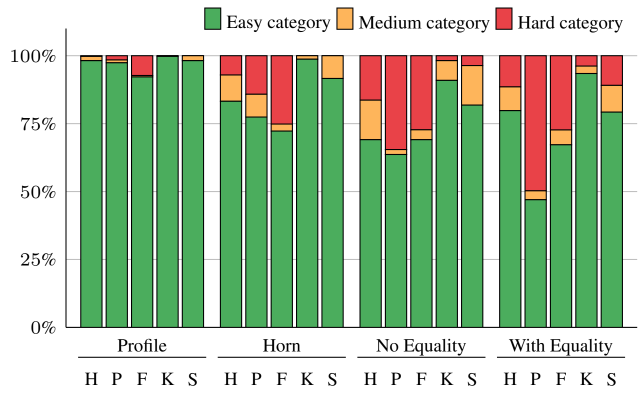


Figure 4: Percentage of Easy, Medium and Hard Ontologies per Ontology Group for HermiT (H), Pellet (P), FaCT++ (F), Konclude (K), and Sequoia (S)

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.

5 Evaluation

We have implemented our calculus in a prototype system called Sequoia. The calculus was implemented exactly as presented in this paper, with no optimisation other than a suitable indexing scheme for clauses. The system is written in Scala, and it can be used via the command line or the OWL API. It currently handles the $\mathcal{SRIQ}$ subset of OWL 2 DL (i.e., it does not support datatypes, nominals, or reflexive roles), for which it supports ontology classification and concept satisfiability; other standard services such as ABox realisation are currently not supported.

We have evaluated our system using the methodology by Steigmiller, Liebig, and Glimm (2014) by comparing Sequoia with HermiT 1.3.8, Pellet 2.3.1, FaCT++ 1.6.4, and Konclude 1.6.1. We used all reasoners in single-threaded mode in order to compare the underlying calculi; moreover, Sequoia was configured to use the cautious strategy. All systems, ontologies, and test results are available online.[1]

We used the Oxford Ontology Repository[2] from which we excluded 7 ontologies with irregular RBoxes. Since Sequoia does not support datatypes or nominals, we have systematically replaced datatypes and nominals with fresh classes and data properties with object properties, and we have removed ABox assertions. We thus obtained a corpus of 777 ontologies on which we tested all reasoners.

We run 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.

Figure 3 shows an overview of the classification times for the entire corpus. The y-axis shows the classification times in 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 reasoner's curve means that the 50th easiest ontology for that reasoner took 100 ms to classify.

Sequoia could process most ontologies (733 out of 784) in under 10s, which is consistent with the other reasoners. The system was fairly robust, failing on only 22 ontologies; in contrast, HermiT failed on 42, Pellet on 138, FaCT++ on 132, and Konclude on 8 ontologies. Moreover, Sequoia succeeded on 21 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 number restrictions.

Figure 4 shows an overview of how each reasoner performed on each type of ontology. We partitioned the ontologies in 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 number restrictions; and disjunctive and with number restrictions. 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), 'medium' (10s to 5min), and '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

[1]http://krr-nas.cs.ox.ac.uk/2015/KR/cr/

[2]http://www.cs.ox.ac.uk/isg/ontologies/

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 equality, 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 a nice pay-as-you-go behaviour. Furthermore, problematic ontologies seem to mostly contain complex RBoxes or large numbers in cardinality restrictions, which suggests promising directions for future optimisation.

6 Conclusion and Future Work

We have presented the first consequence based calculus for $\mathcal{SRIQ}$—a DL that includes both disjunction and counting quantifiers. Our calculus combines ideas from state of the art resolution and (hyper)tableau calculi, including the use of ordered paramodulation for equality reasoning. Despite its increased complexity, the calculus mimics existing calculi on $\mathcal{ELH}$ ontologies. Although it is an early prototype with plenty of room for optimisation, our system Sequoia is competitive with well-established reasoners and it exhibits nice pay-as-you-go behaviour in practice.

For future work, we are confident that we can extend the calculus to support role reflexivity and datatypes, thus handling all of OWL 2 DL except nominals. In contrast, handling nominals seems to be much more involved. In fact, adding nominals to $\mathcal{ALCHIQ}$ raises the complexity of reasoning to NExpTime so a worst-case optimal calculus must be nondeterministic, which is quite different from all consequence-based calculi we are aware of. Moreover, a further challenge is to modify the calculus so that it can effectively deal with large numbers in number restrictions.

References

Baader, F., and Nipkow, T. 1998. *Term Rewriting and All That*. Cambridge University Press.

Baader, F.; Calvanese, D.; McGuinness, D.; Nardi, D.; and Patel-Schneider, P. F., eds. 2003. *The Description Logic Handbook: Theory, Implementation and Applications*. Cambridge University Press.

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

Bachmair, L., and Ganzinger, H. 2001. Resolution Theorem Proving. In Robinson, A., and Voronkov, A., eds., *Handbook of Automated Reasoning*, volume I. Elsevier Science. chapter 2, 19–99.

Bate, A.; Motik, B.; Cuenca Grau, B.; Simančík, F.; and Horrocks, I. 2016. Extending Consequence-Based Reasoning to $\mathcal{SRIQ}$. arXiv:1602.04498 [cs.AI].

Glimm, B.; Horrocks, I.; Motik, B.; Stoilos, G.; and Wang, Z. 2014. HermiT: An OWL 2 Reasoner. *Journal of Automated Reasoning* 53(3):245–269.

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

Kazakov, Y. 2008. $\mathcal{RIQ}$ and $\mathcal{SROIQ}$ are Harder than $\mathcal{SHOIQ}$. In Brewka, G., and Lang, J., eds., *Proc. of the 11th Int. Joint Conf. on Principles of Knowledge Representation and Reasoning (KR 2008)*, 274–284. Sydney, NSW, Australia: AAAI Press.

Kazakov, Y. 2009. Consequence-Driven Reasoning for Horn SHIQ Ontologies. In Boutilier, C., ed., *Proc. of the 21st Int. Joint Conf. on Artificial Intelligence (IJCAI 2009)*, 2040–2045.

Motik, B.; Shearer, R.; and Horrocks, I. 2009. Hypertableau Reasoning for Description Logics. *Journal of Artificial Intelligence Research* 36:165–228.

Nieuwenhuis, R., and Rubio, A. 1995. Theorem Proving with Ordering and Equality Constrained Clauses. *Journal of Symbolic Computation* 19(4):312–351.

Ortiz, M.; Rudolph, S.; and Simkus, M. 2010. Worst-Case Optimal Reasoning for the Horn-DL Fragments of OWL 1 and 2. In Lin, F.; Sattler, U.; and Truszczynski, M., eds., *Proc. of the 12th Int. Conf. on Knowledge Representation and Reasoning (KR 2010)*, 269–279. Toronto, ON, Canada: AAAI Press.

Riazanov, A., and Voronkov, A. 2002. The design and implementation of VAMPIRE. *AI Communications* 15(2–3):91–110.

Schulz, S. 2002. E—A Brainiac Theorem Prover. *AI Communications* 15(2–3):111–126.

Simančík, F.; Kazakov, Y.; and Horrocks, I. 2011. Consequence-Based Reasoning beyond Horn Ontologies. In Walsh, T., ed., *Proc. of the 22nd Int. Joint Conf. on Artificial Intelligence (IJCAI 2011)*, 1093–1098.

Simančík, F.; Motik, B.; and Horrocks, I. 2014. Consequence-Based and Fixed-Parameter Tractable Reasoning in Description Logics. *Artificial Intelligence* 209:29–77.

Sirin, E.; Parsia, B.; Cuenca Grau, B.; Kalyanpur, A.; and Katz, Y. 2007. Pellet: A practical OWL-DL reasoner. *Journal of Web Semantics* 5(2):51–53.

Steigmiller, A.; Glimm, B.; and Liebig, T. 2014. Coupling Tableau Algorithms for Expressive Description Logics with Completion-Based Saturation Procedures. In Demri, S.; Kapur, D.; and Weidenbach, C., eds., *Proc. of the 7th Int. Joint Conf. on Automated Reasoning (IJCAR 2014)*, volume 8562 of *LNCS*, 449–463. Vienna, Austria: Springer.

Steigmiller, A.; Liebig, T.; and Glimm, B. 2014. Konclude: System description. *Journal of Web Semantics* 27:78–85.

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