Consequence-based Reasoning for Description Logics with Disjunction, Inverse Roles, Number Restrictions, and Nominals

David Tena Cucala, Bernardo Cuenca Grau and **Ian Horrocks**
Department of Computer Science, University of Oxford, UK
david.tena.cucala@cs.ox.ac.uk, bernardo.cuenca.grau@cs.ox.ac.uk, ian.horrocks@cs.ox.ac.uk

Abstract

We present a consequence-based calculus for concept subsumption and classification in the description logic $\mathcal{ALCHOIQ}$, which extends $\mathcal{ALC}$ with role hierarchies, inverse roles, number restrictions, and nominals. By using standard transformations, our calculus extends to $\mathcal{SROIQ}$, which covers all of OWL 2 DL except for datatypes. A key feature of our calculus is its pay-as-you-go behaviour: unlike existing algorithms, our calculus is worst-case optimal for all the well-known proper fragments of $\mathcal{ALCHOIQ}$, albeit not for the full logic.

1 Introduction

Description logics (DLs) [Baader *et al.*, 2003] are a family of knowledge representation formalisms which are widely used in applications. Although the basic DL reasoning problems, such as concept satisfiability and subsumption, are of high worst-case complexity for expressive DLs, different calculi have been developed and implemented in practical systems. Tableau and hyper-tableau calculi [Baader and Sattler, 2001; Motik *et al.*, 2009] are a prominent reasoning technique underpinning many DL reasoners [Sirin *et al.*, 2007; Tsarkov and Horrocks, 2006; Haarslev *et al.*, 2012; Glimm *et al.*, 2014; Steigmiller *et al.*, 2014]. To check whether a concept subsumption relationship holds, (hyper-)tableau calculi attempt to construct a finite representation of an ontology model disproving the given subsumption. The constructed models can, however, be large—a common source of performance issues; this problem is exacerbated in classification tasks due to the large number of subsumptions to be tested.

Another major category of DL reasoning calculi comprises methods based on first-order logic resolution [Bachmair and Ganzinger, 2001]. A common approach to ensure both termination and worst-case optimal running time is to parametrise resolution to ensure that the calculus only derives a bounded number of clauses [Nivelle *et al.*, 2000; Hustadt and Schmidt, 2002; Schmidt and Hustadt, 2013; Hustadt *et al.*, 2008; Ganzinger and De Nivelle, 1999; Kazakov and Motik, 2006; Hustadt *et al.*, 2004]. This technique has been implemented for instance, in the KAON2 reasoner for $\mathcal{SHIQ}$. Resolution can also be used to simulate model-building (hyper)tableau techniques [Hustadt and Schmidt, 1999], including blocking methods which ensure termination [Georgieva *et al.*, 2003].

Consequence-based (CB) calculi have emerged as a promising approach to DL reasoning combining features of (hyper)tableau and resolution [Baader *et al.*, 2005; Kazakov, 2009; Kazakov *et al.*, 2012; Bate *et al.*, 2016]. On the one hand, similarly to resolution, they derive formulae entailed by the ontology (thus avoiding the explicit construction of large models), and they are typically worst-case optimal. On the other hand, clauses are organised into *contexts* arranged as a graph structure reminiscent to that used for model construction in (hyper)tableau; this prevents CB calculi from drawing many unnecessary inferences and yields a nice goal-oriented behaviour. Furthermore, in contrast to both resolution and (hyper)tableau, CB calculi can verify a large number of subsumptions in a single execution, allowing for one-pass classification. Finally, CB calculi are very practical and systems based on them have shown outstanding performance.

CB calculi were first proposed for the $\mathcal{EL}$ family of DLs [Baader *et al.*, 2005; Kazakov *et al.*, 2012], and later extended to more expressive logics like Horn-$\mathcal{SHIQ}$ [Kazakov, 2009], Horn-$\mathcal{SROIQ}$ [Ortiz *et al.*, 2010], and $\mathcal{ALCH}$ [Simančík *et al.*, 2011]. A unifying framework for CB reasoning was developed in [Simančík *et al.*, 2014] for $\mathcal{ALCHI}$, introducing the notion of *contexts* as a mechanism for constraining resolution inferences and making them goal-directed. The framework has been extended to the DLs $\mathcal{ALCHIQ}$, which supports number restrictions and inverse roles [Bate *et al.*, 2016]; $\mathcal{ALCHIO}$, which supports inverse roles and nominals [Tena Cucala *et al.*, 2017], and $\mathcal{ALCHOQ}$, supporting nominals and number restrictions [Karahroodi and Haarslev, 2017].

To the best of our knowledge, however, no CB calculus can handle DLs supporting simultaneously all Boolean connectives, inverse roles, number restrictions, and nominals. Such DLs, which underpin the standard ontology languages, pose significant challenges for consequence-based reasoning. Indeed, DLs lacking inverse roles, number restrictions, or nominals enjoy a variant of the *forest model property*, which is exploited by reasoning algorithms. However, no such property holds when a DL simultaneously supports all the aforementioned features; for non-Horn DLs, this results in a complexity jump from ExpTime to NExpTime, and complicates the design of reasoning calculi [Horrocks and Sattler, 2005].

In this paper we present the first consequence-based cal-

culus for the DL $\mathcal{ALCHOIQ}$, which supports all Boolean connectives, role hierarchies, inverse roles, number restrictions, and nominals. By using well-known transformations, our calculus extends to $\mathcal{SROIQ}$, which covers OWL 2 DL except for datatypes [Horrocks *et al.*, 2006]. Following Bate *et al.* [2016], we encode derived consequences as first-order clauses of a specific form and handle equality reasoning using a variant of ordered paramodulation. To account for nominals we allow for ground atoms in derived clauses and group consequences about named individuals into a *root context*. We have carefully crafted the rules of our calculus so that it exhibits worst-case optimal performance for the known proper fragments of $\mathcal{ALCHOIQ}$. In particular, it works in deterministic exponential time for all of $\mathcal{ALCHOI}$, $\mathcal{ALCHOQ}$, $\mathcal{ALCHIQ}$ and Horn-$\mathcal{ALCHOIQ}$, and in polynomial time for the lightweight DL $\mathcal{ELHO}$. Our calculus is, however, not worst-case optimal for the full logic $\mathcal{ALCHOIQ}$, exhibiting similar worst-case running time as other well-known calculi for expressive DLs [Motik *et al.*, 2009; Kazakov and Motik, 2006]. Nevertheless, the source of additional complexity is very localised and only manifests when disjunction, nominals, number restrictions and inverse roles interact simultaneously—a rare situation in practice. Although our results are theoretical, we believe that our calculus can be seamlessly implemented as an extension of the $\mathcal{SRIQ}$ reasoner Sequoia [Bate *et al.*, 2016]. All proofs are presented in an extended version of the paper [Tena Cucala *et al.*, 2018].

2 Preliminaries

Many-sorted clausal equational logic. We use standard terminology for many-sorted first-order logic with equality ($\approx$) as the only predicate. This is w.lo.g. since predicates other than equality can be encoded by means of an entailment-preserving transformation [Nieuwenhuis and Rubio, 2001].

A many-sorted signature Σ is a pair $\langle \Sigma^S, \Sigma^F \rangle$ with Σ^S a non-empty set of sorts, and Σ^F a countable set of function symbols. Each $f \in \Sigma^F$ is associated to a symbol type, which is an $n+1$-tuple $\langle \mathsf{s}_1, \dots, \mathsf{s}_{n+1} \rangle$ with each $\mathsf{s}_i \in \Sigma^S$. The sort of f is s_{n+1} and its arity is n; if $n = 0$, then f is a constant. For each $\mathsf{s} \in \Sigma^S$, let X_s be a disjoint, countable set of variables. The set of terms is the smallest set containing all variables in X_s as terms of sort s, and all expressions $f(t_1, \dots, t_n)$ as terms of sort s_{n+1}, where each t_i is a term of sort s_i. A term is ground if it has no variables. We use the standard definition of a position p of a term t as an integer string identifying an occurrence of a subterm $t|_p$, as well as the standard notion of a substitution, represented as an expression $\{x_1 \mapsto t_1, \dots, x_n \mapsto t_n\}$ containing all non-identity mappings. We represent by $t[r]_p$ the result of replacing the subterm in position p of t by a term r of the same sort.

An equality is an expression $s \approx t$ with s and t terms of the same sort. An inequality is of the form $\neg(s \approx t)$, and is written as $s \not\approx t$. A literal is an equality or an inequality. A clause is a sentence $\forall \vec{x}\ (\Gamma \to \Delta)$, with the body Γ a conjunction of equalities, the head Δ a disjunction of literals, and $\vec{x}$ the variables occurring in the clause. The quantifier is often omitted, conjunctions and disjunctions are treated as sets, and the empty conjunction (disjunction) is written as $\top$ ($\bot$).

Let HU^s be the set of ground terms of sort s. A Herbrand equality interpretation $\mathcal{I}$ is a set of ground equalities satisfying the usual properties of equality: (i) reflexivity, (ii) symmetry, (iii) transitivity, and (iv) $t[s]_p \approx t \in \mathcal{I}$ whenever $t|_p \approx s \in \mathcal{I}$, for each ground term t, position p, and ground term s. For any ground conjunction, ground disjunction, (not necessarily ground) clause, or a set thereof; an interpretation $\mathcal{I}$ satisfies it according to the usual criteria, with the difference that each quantified variable of sort s ranges only over HU^s. We write $\mathcal{I} \models \alpha$ if $\mathcal{I}$ satisfies α, and say that $\mathcal{I}$ is a *model* of α. Entailment is defined as usual.

Orders A strict (non-total) order $\succ$ on a non-empty set S is a binary, irreflexive, and transitive relation between elements of S. A strict order induces a non-strict order $\succeq$ by taking the reflexive closure of $\succ$. A total order $>$ is a strict order such that for any $a, b \in S$, either $a > b$ or $b > a$. For any of these orders $\circ$, we write $a \circ N$, if $a \circ b$ for each $b \in N$, where $a \in S$, $N \subseteq S$. The multiset extension of $\circ$ is defined as follows: for S-multisets M and N, we have $M \circ N$ iff for each $a \in N \backslash M$, there is $b \in M \backslash N$ such that $b \circ a$, where $\backslash$ is the multiset difference operator. Order $\circ$ induces an order between literals by treating each equality $s \approx t$ as the multiset $\{s, t\}$, and each inequality $s \not\approx t$ as the multiset $\{s, s, t, t\}$.

Description Logics DL expressions can be transformed into clauses of many-sorted equational logic in a way that preserves satisfiability and entailment [Schmidt and Hustadt, 2007]. Following standard practice, we use a two-sorted signature with a sort a representing standard FOL terms, and a sort p for standard FOL atoms. The set of function symbols is the disjoint union of a set Σ_B of atomic concepts B_i of type $\langle \mathsf{a}, \mathsf{p} \rangle$, a set Σ_S of atomic roles S_i of type $\langle \mathsf{a}, \mathsf{a}, \mathsf{p} \rangle$, a set Σ_f of functions f_j of type $\langle \mathsf{a}, \mathsf{a} \rangle$, and a set Σ_o of named individuals o of type $\langle \mathsf{a} \rangle$. A term of the form $f_j(t)$ is an f_j successor of t, and t is its predecessor. Our signature uses variables $\{x\} \cup \{z_i\}_{i \geq 1}$ of sort a, where x is called a central variable, and each z_i is a neighbour variable. A DL-a-term is a term of the form $z_i, x, f_i(x)$, or o. A DL-p-term is of the form $B_i(z_j)$, $B_i(x)$, $B_i(f_j(x))$, $B_i(o)$, $S_i(z_j, x)$, $S_i(x, z_j)$, $S_i(x, f_j(x))$, $S_i(f_j(x), x)$, $S_i(o, x)$ or $S_i(x, o)$. A DL-literal is either an equality of the form $A \approx \mathsf{true}$ (or just A) with A a DL-p-term, or an (in)equality between DL-a-terms. A DL-clause contains only body atoms of the form $B_i(x), S_i(z_j, x)$, or $S_i(x, z_j)$, and only DL-literals in the head. To ensure completeness and termination of our calculus, we require that each z_j in the head occurs also in the body and that if the body contains two or more neighbour variables z_j, then the clause is of the form DL4 in Table 1. An *ontology* is a finite set of DL-clauses.

An ontology is $\mathcal{ALCHOIQ}$ if each DL-clause is of the form given in Table 1. An $\mathcal{ALCHOIQ}$ ontology is $\mathcal{ALCHOI}$ if it does not contain axioms DL4 and all axioms DL2 satisfy $n = 1$; it is $\mathcal{ALCHOQ}$ if it does not contain axioms DL6; it is $\mathcal{ALCHIQ}$ if it does not contain axioms DL7-DL8. Furthermore, it is Horn if each axiom DL1 satisfies $n \leq m \leq n+1$ and each axiom DL4 satisfies $n = 1$.

DL1	$\prod_{1\le i\le n} B_i \sqsubseteq \bigsqcup_{n+1\le i\le m} B_i$	$\rightsquigarrow$	$\bigwedge_{1\le i\le n} B_i(x) \rightarrow \bigvee_{n+1\le i\le m} B_i(x)$
DL2	$B_1 \sqsubseteq \geq nS.B_2$	$\rightsquigarrow$	$B_1(x) \rightarrow B_2(f_i(x)),\ 1 \le i \le n$ $B_1(x) \rightarrow S(x, f_i(x)),\ 1 \le i \le n$ $B_1(x) \rightarrow f_i(x) \not\approx f_j(x), 1 \le i < j \le n$
DL3	$\exists S.B_1 \sqsubseteq B_2$	$\rightsquigarrow$	$S(z,x) \wedge B_1(x) \rightarrow B_2(z)$
DL4	$B_1 \sqsubseteq \leq nS.B_2$	$\rightsquigarrow$	$S(z_1,x) \wedge B_2(x) \rightarrow S_{B_2}(z_1,x)$ $B_1(x) \wedge \bigwedge_{1\le i\le n+1} S_{B_2}(x,z_i) \rightarrow \bigvee_{1\le i<j\le n+1} z_i \approx z_j$
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	$\{o\} \sqsubseteq B_1$	$\rightsquigarrow$	$\top \rightarrow B_1(o)$
DL8	$B_1 \sqsubseteq \{o\}$	$\rightsquigarrow$	$B_1(x) \rightarrow x \approx o$

Table 1: DL axioms as clauses. Roles S_{B_2} in DL4 are fresh.

Finally, it is in $\mathcal{ELHO}$ if it is Horn and contains only axioms DL1, DL2 with $n = 1$, DL3, DL5, DL7, or DL8.

3 A CB Algorithm for $\mathcal{ALCHOIQ}$

Consequence-based reasoning combines features of both (hyper-)tableau calculi and resolution. Although the presentation of CB calculi varies in the literature, all CB calculi that we know of share certain core characteristics. First, they derive in a single run all consequences of a certain form (typically subsumptions between atomic concepts, $\top$ and $\bot$) and hence they are not just refutationally complete. Second, like resolution, they compute a saturated set of clauses—represented either as DL-style axioms [Baader *et al.*, 2005; Kazakov, 2009; Simančík *et al.*, 2011; 2014] or using first-order notation [Bate *et al.*, 2016; Tena Cucala *et al.*, 2017; Karahroodi and Haarslev, 2017]—the shape of which is restricted to ensure termination. Third, unlike resolution, where all clauses are kept in a single set, CB calculi construct a graph-like *context structure* where clauses can only interact with other clauses in the same context or in neighbouring contexts, thus guiding the reasoning process in a way that is reminiscent of (hyper-)tableau calculi. Fourth, the expansion of the context structure during a run of the algorithm is determined by an *expansion strategy*, which controls when and how to create or reuse contexts.

In the remainder of this section we define our CB calculus, and specify a reasoning algorithm based on it. We then establish its key correctness and complexity properties.

3.1 Definition of the Calculus

Throughout this section we fix an arbitrary ontology $\mathcal{O}$, and we let $\Sigma_f^{\mathcal{O}}$, $\Sigma_B^{\mathcal{O}}$ and $\Sigma_S^{\mathcal{O}}$ be the sets of functions, atomic concepts and atomic roles in $\mathcal{O}$, respectively; all our definitions and theorems are implicitly relative to $\mathcal{O}$.

The set of *nominal labels* Π for $\mathcal{O}$ is the smallest set containing the empty string and every string ρ of the form $S_1^{j_1} \cdot \ldots \cdot S_n^{j_n}$, with $j_k \in \mathbb{N}$, and $S_i \in \Sigma_S$. The set of named individuals $\Sigma_o^{\mathcal{O}}$ for $\mathcal{O}$ is then defined as $\{o_\rho \mid o \text{ individual in } \mathcal{O}, \rho \in \Pi\}$. Intuitively, the set $\Sigma_o^{\mathcal{O}}$ consists of the individuals occurring explicitly in $\mathcal{O}$ plus a set of *additional nominals*, the introduction of which is reminiscent of existing (hyper-)tableau calculi [Horrocks and Sattler, 2005; Motik *et al.*, 2007].

Following Bate *et al.* [2016], our calculus for $\mathcal{ALCHOIQ}$ represents all derived consequences in contexts as *context clauses* in many-sorted equational logic, rather than DL-style axioms. Context clauses use only variables x and y, which carry a special meaning. Intuitively, a context represents a set of similar elements in a model of the ontology; when variable x corresponds to such an element, y corresponds to its predecessor, if it exists. This naming convention determines rule application in the calculus, and should be distinguished from variables x and z_i in DL-clauses, where the latter can map to both predecessors and successors of the elements assigned to x. Context clauses are defined analogously to [Bate *et al.*, 2016], where the main difference is that we allow context literals mentioning named individuals; furthermore, our calculus defines a distinguished *root context* where most inferences involving such literals take place. This context represents the non tree-like part of the model, and it exchanges information with other contexts using newly devised inference rules.

Definition 1. A *context* a*-term* is a term of sort a which is either x, or y, or a named individual $o \in \Sigma_o^{\mathcal{O}}$, or of the form $f(x)$ for $f \in \Sigma_f^{\mathcal{O}}$. A *context* p*-term* is a term of sort p of the form $B(y)$, $B(x)$, $B(f(x))$, $B(o)$, $S(x,y)$, $S(y,x)$, $S(x,x)$, $S(x,f(x))$, $S(f(x),x)$, $S(x,o)$, $S(o,x)$, $S(o,o')$, for $f \in \Sigma_f^{\mathcal{O}}$, o and $o' \in \Sigma_o^{\mathcal{O}}$, $B \in \Sigma_B^{\mathcal{O}}$, and $S \in \Sigma_S^{\mathcal{O}}$. A *root context* a*-term* (p-term) is a term of sort a (p) of the form $t\{x \mapsto o'\}$, with t a context a-term (p-term) and $o' \in \Sigma_o^{\mathcal{O}}$. A (root) *context atom* is an equality of the form $A \approx \mathsf{true}$, written simply as A, with A a context (root) p-term; a (root) *context literal* is a (root) context atom, an inequality $\mathsf{true} \not\approx \mathsf{true}$, or an equality or inequality between a-terms (replacing x by $o' \in \Sigma_o^{\mathcal{O}}$).

A (root) *context clause* is a clause of (root) context atoms in the body and (root) context literals in the head. A *query* clause has only atoms of the form $B(x)$, with $B \in \Sigma_B^{\mathcal{O}}$.

The kinds of information to be exchanged between adjacent contexts is determined by a set of *triggers*, which are named after the rules that they activate.

Definition 2. The set of *successor triggers* Su is the smallest set of atoms satisfying the following properties for each clause $\Gamma \rightarrow \Delta$ in $\mathcal{O}$: *(i)* $B(x) \in \Gamma$ implies $B(y) \in \mathsf{Su}$; *(ii)* $S(x,z_i) \in \Gamma$ implies $S(x,y) \in \mathsf{Su}$; and *(iii)* $S(z_i,x) \in \Gamma$ implies $S(y,x) \in \mathsf{Su}$. The set of *predecessor triggers* Pr is defined as the set of literals

$$\{A\{x \mapsto y,\ y \mapsto x\} \mid A \in \mathsf{Su}\} \cup \{B(y) \mid B \in \Sigma_B^{\mathcal{O}}\} \cup \{x \approx y\} \cup \{x \approx o \mid o \in \Sigma_o^{\mathcal{O}}\} \cup \{y \approx o \mid o \in \Sigma_o^{\mathcal{O}}\}.$$

The set of *root successor triggers* Su^r consists of all atoms $B(o)$, $S(y,o)$ and $S(o,y)$ with $B \in \Sigma_B^{\mathcal{O}}$, $S \in \Sigma_S^{\mathcal{O}}$ and $o \in \Sigma_o^{\mathcal{O}}$. The set of *root predecessor triggers* Pr^r consists of $\mathsf{Su}^r \cup \{B(y) \mid B \in \Sigma_B^{\mathcal{O}}\} \cup \{y \approx o \mid o \in \Sigma_o^{\mathcal{O}}\}$.

The definition of triggers extends that in [Bate *et al.*, 2016] by considering equalities of a variable and an individual as information that should be propagated to predecessor contexts, and by identifying a specific set of triggers for propagating information to and from the distinguished root context.

Same as in resolution and other CB calculi, clauses are ordered using a term order $\succ$ based on a total order $\gg$ on function symbols of sort a. The order restricts the derived clauses since only $\succ$-maximal literals can participate in inferences.

The following definition specifies the conditions that $\succ$ must satisfy; although each context can use a different $\succ$ order, a-terms are compared in the same way across all contexts since $\succcurlyeq$ is globally defined. In [Tena Cucala *et al.*, 2018, Appendix A] we show how to construct a context order once $\succcurlyeq$ is fixed.

Definition 3. Let $\succcurlyeq$ be a total order on symbols of $\Sigma_f^{\mathcal{O}}$ and $\Sigma_o^{\mathcal{O}}$ such that for every $\rho \in \Pi$, if $\rho = \rho' \cdot \rho''$, then $o_\rho \succcurlyeq o_{\rho'}$. A (root) *context order* $\succ$ w.r.t. $\succcurlyeq$ is a strict order on (root) context atoms satisfying each of the following properties:

1. $A \succ x \succ y \succ \mathsf{true}$ for each context p-term $A \neq \mathsf{true}$;
2. $n \succ m$ for each pair $n, m \in \Sigma_o^{\mathcal{O}}$ with $n \succcurlyeq m$;
3. $f(x) \succ g(x)$, for all $f, g \in \Sigma_f^{\mathcal{O}}$ with $f \succcurlyeq g$;
4. $t[s_1]_p \succ t[s_2]_p$ for any context term t, position p, and context terms s_1, s_2 such that $s_1 \succ s_2$;
5. $s \succ s|_p$ for each context term s and proper position p in s;
6. $A \not\succ s$ for each atom $A \approx \mathsf{true} \in \mathsf{Pr}$ (Pr^r) and context term $s \notin \{x, y, \mathsf{true}\} \cup \Sigma_o^{\mathcal{O}}$.

The main difference between Definition 3 and the orderings used in prior work is the additional requirement that the global order $\succcurlyeq$ must satisfy on the set of nominals; this is necessary to ensure both completeness and termination.

We use a notion of redundancy elimination analogous to that of prior work to significantly reduce the amount of clauses derived by the algorithm.

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. $t \approx t \in \Delta$ or $\{t \approx s, t \not\approx s\} \subseteq \Delta$ for some a-term t, s, or
2. $\Gamma' \to \Delta' \in U$ for some $\Gamma' \subseteq \Gamma$ and $\Delta' \subseteq \Delta$.

The first condition in Definition 4 captures tautological statements, whereas the second condition captures clause subsumption. Similarly to prior work, clauses $A \to A$ are not deemed tautological in our calculus since they imply that atom A may hold in a context.

We next define the notion of a context structure $\mathcal{D}$ as a digraph. Each node v, labelled with a set of clauses $\mathcal{S}_v$, represents a set of "similar" terms in a model of $\mathcal{O}$; edges, labelled with a function symbol, represent connections between neighboring contexts. Each context v is assigned a core core_v specifying the atoms that must hold for all terms in the canonical model described by the context, and a term order $\succ_v$ that restricts the inferences applicable to $\mathcal{S}_v$; since core_v holds implicitly in a context, the conjunction core_v is not included in the body of any clause in $\mathcal{S}_v$.

Definition 5. A context structure for $\mathcal{O}$ is a tuple $\mathcal{D} = \langle \mathcal{V}, \mathcal{E}, \mathsf{core}, \mathcal{S}, \succcurlyeq, \succ \rangle$ where $\mathcal{V}$ is a finite set of *contexts* containing the *root context* v_r; $\mathcal{E}$ is a subset of $\mathcal{V} \times \mathcal{V} \times \Sigma_f^{\mathcal{O}}$; core is a function mapping each context v to a conjunction core_v of atoms of the form $B(x)$, $S(x, y)$, $S(y, x)$; $\mathcal{S}$ is a function mapping each non-root context v to a set of context clauses and v_r to a set of root context clauses, $\succcurlyeq$ is a total order and $\succ$ is a function mapping each (root) context v to a (root) context order $\succ_v$ w.r.t. $\succcurlyeq$ where all $\succcurlyeq$ and $\succ_v$ satisfy Definition 3.

We now define when a context structure is sound with respect to $\mathcal{O}$. In prior work, soundness was defined by requiring that clauses derived by the calculus are logical consequences of $\mathcal{O}$. Our calculus, however, introduces additional nominals, and clauses mentioning them are not logical consequences of $\mathcal{O}$. Furthermore, not all the the additional nominals generated by our calculus will correspond to actual elements of a canonical model. To address this difficulty, we introduce the following notions of N-reduction and N-compatibility.

Definition 6. Let N be a (possibly empty) set of additional nominals in $\Sigma_o^{\mathcal{O}}$. Interpretation $\mathcal{I}$ with domain $\Delta^{\mathcal{I}}$ is N-compatible if the following conditions hold for each non-empty nominal label ρ, which we rewrite as $\rho = \rho' \cdot S$:

- if $o_\rho \in N$, $\mathcal{I} \not\models S(o_{\rho'}, u)$ for each $u \in \Delta^{\mathcal{I}}$,
- if $o_\rho \notin N$, $\mathcal{I} \models S(o_{\rho'}, o_\rho)$, and there is $k_0 \in \mathbb{N}$ s.t. for each $u \in \Delta^{\mathcal{I}}$, $\mathcal{I} \models S(o_{\rho'}, u)$ implies $\mathcal{I} \models u \approx o_{\rho' \cdot S^k}$ for some $k \leq k_0$, and for each $k > k_0$, $\mathcal{I} \models o_{\rho' \cdot S^k} \approx o_{\rho' \cdot S^1}$'.

The N-reduction $N(\Gamma \to \Delta)$ of a clause $\Gamma \to \Delta$ is empty if an element of N occurs in Γ or in an inequality in Δ, and the clause $\Gamma \to N(\Delta)$ otherwise, with $N(\Delta)$ obtained from Δ by removing all literals mentioning a term in N.

Intuitively, given N and a model $\mathcal{I}$ of $\mathcal{O}$, the notion of N-reduction tests whether it is possible to map the additional nominals not in N occurring in derived clauses to actual domain elements of $\mathcal{I}$ without invalidating the model. This intuition leads to the following notion of soundness.

Definition 7. A context structure $\mathcal{D} = \langle \mathcal{V}, \mathcal{E}, \mathsf{core}, \mathcal{S}, \succcurlyeq, \succ \rangle$ is *sound* if, for every model $\mathcal{I}$ of $\mathcal{O}$, there exists a (possibly empty) set N of additional nominals and an N-compatible conservative extension $\mathcal{J}$ of $\mathcal{I}$ satisfying the following clauses: *(i)* $N(\mathsf{core}_v \wedge \Gamma \to \Delta)$ for each $v \in \mathcal{V}$ and each $\Gamma \to \Delta$ in $\mathcal{S}_v$; and *(ii)* $N(\mathsf{core}_u \to \mathsf{core}_v\{x \mapsto f(x), y \mapsto x\})$ for each $\langle u, v, f \rangle \in \mathcal{E}$.

As in existing CB calculi, the rules of our calculus are parameterised by an *expansion strategy* used to decide whether to create new contexts or re-use already existing ones.

Definition 8. An expansion strategy strat is a polynomially computable function which takes as input a triple $(f, K_1, \mathcal{D})$, where $f \in \Sigma_f^{\mathcal{O}}$, $K_1 \subseteq \mathsf{Su}$, $\mathcal{D} = \langle \mathcal{V}, \mathcal{E}, \mathcal{S}, \mathsf{core}, \succcurlyeq, \succ \rangle$ is a context structure, and returns a triple $(v, \mathsf{core}, \succ)$ such that $\mathsf{core} \subseteq K_1$, $\succ$ is a context order w.r.t $\succcurlyeq$, and either $v \notin \mathcal{V}$ or otherwise $v \neq v_r$, $\mathsf{core} = \mathsf{core}_v$ and $\succ = \succ_v$.

Three expansion strategies are typically considered in practice (see [Simančík *et al.*, 2011] for details). The *trivial* strategy pushes all inferences to a single context $v_\top$ with empty core and always returns $(v_\top, \top)$. The *cautious* strategy only creates contexts for concept names in existential restrictions; it returns $(v_\top, \top)$ unless f occurs in $\mathcal{O}$ in exactly one atom $B(f(x))$ with $B \in \Sigma_B^{\mathcal{O}}$ and $B(x) \in K_1$, in which case it returns $(v_B, B(x))$. Finally, the *eager* strategy creates a new context for each conjunction K_1 by returning (v_{K_1}, K_1).

The inference rules of our calculus are specified in Tables 2 and 3. As in prior work, a rule is not triggered if the clauses that would be derived are already contained up to redundancy in the corresponding contexts. Rules in Table 2 are a simple generalisation of those in [Bate *et al.*, 2016] for $\mathcal{ALCHIQ}$

Core	If 1. $A \in \mathsf{core}_v$ then add $\top \to A$ to $\mathcal{S}_v$.
Hyper	If 1. $\bigwedge_{i=1}^{n} A_i \to \Delta \in \mathcal{O}$ and $\sigma(x)=x$ if $v \neq v_r$ or $\sigma(x) \in \Sigma_o^{\mathcal{O}}$ o.w. 2. and $\Gamma_i \to \Delta_i \vee A_i\sigma \in \mathcal{S}_v$ with $\Delta_i \not\succeq_v A_i\sigma$, for $1 \leq i \leq n$, then add $\bigwedge_{i=1}^{n} \Gamma_i \to \bigvee_{i=1}^{n} \Delta_i \vee \Delta\sigma$ to $\mathcal{S}_v$.
Eq	If 1. $\Gamma_1 \to \Delta_1 \vee s_1 \approx t_1 \in \mathcal{S}_v$ with $t_1 \not\succeq_v s_1$ and $\Delta_1 \not\succeq_v s_1 \approx t_1$, 2. $\Gamma_2 \to \Delta_2 \vee s_2 \bowtie t_2 \in \mathcal{S}_v$ with $\bowtie \in \{\approx, \not\approx\}$, $t_2 \not\succ_v s_2$, $\Delta_2 \not\succeq_v s_2 \bowtie t_2$, and $s_2|_p$ is not a variable, and if $s_2|_p \in \Sigma_o^{\mathcal{O}}$, then s_2 contains no function symbols. 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 1. $\Gamma \to \Delta \vee t \not\approx t \in \mathcal{S}_v$, then add $\Gamma \to \Delta$ to $\mathcal{S}_v$.
Fact	If 1. $\Gamma \to \Delta \vee s \approx t \vee s \approx t' \in \mathcal{S}_v$, 2. with $\Delta \cup \{s \approx t\} \not\succeq_v s \approx t'$ and $t' \not\succ_v s$, then add $\Gamma \to \Delta \vee t \not\approx t' \vee s \approx t'$ to $\mathcal{S}_v$.
Elim	If 1. $\Gamma \to \Delta \in \mathcal{S}_v$ 2. and $\Gamma \to \Delta \,\hat{\in}\, \mathcal{S}_v \backslash (\Gamma \to \Delta)$ then remove $\Gamma \to \Delta$ from $\mathcal{S}_v$.
Pred	If 1. $\bigwedge_{i=1}^{n} A_i \wedge \bigwedge_{i=1}^{m} C_i \to \bigvee_{i=1}^{k} L_i \in \mathcal{S}_v$ for $v \neq v_r$, where each C_i is ground, and each A_i is nonground 2. $L_i \in \mathsf{Pr}$ for each nonground L_i, 3. and there is $\langle u, v, f \rangle \in \mathcal{E}$ such that 4. for each A_i, there is $\Gamma_i \to \Delta_i \vee A_i\sigma \in \mathcal{S}_u$ with $\Delta_i \not\succeq_u A_i\sigma$; 5. $\sigma = \{y \mapsto x, x \mapsto f(x)\}$ if $u \neq v_r$, $\sigma = \{y \mapsto o, x \mapsto f(o)\}$ o.w., then add $\bigwedge_{i=1}^{n} \Gamma_i \wedge \bigwedge_{i=1}^{m} C_i \to \bigvee_{i=1}^{n} \Delta_i \vee \bigvee_{i=1}^{k} L_i\sigma$ to $\mathcal{S}_v$,
Succ	If 1. $\Gamma \to \Delta \vee A \in \mathcal{S}_u$ where $\Delta \not\succeq_u A$ and A contains $f(x)$ if $u \neq v_r$ or $f(o)$ for some $o \in \Sigma_o^{\mathcal{O}}$ o.w., 2. there is no $\langle u, v, f \rangle \in \mathcal{E}$ s.t. $A' \to A' \,\hat{\in}\, \mathcal{S}_v \; \forall \; A' \in K_2 \backslash \mathsf{core}_v$ then 1. let $\langle v, \mathsf{core}', \succ' \rangle = \mathsf{strat}(f, K_1, \mathcal{D})$ and if $v \notin \mathcal{V}$, then let 2. $\mathcal{V} = \mathcal{V} \cup \{v\}$, and $\mathsf{core}_v = \mathsf{core}'$, $\succ_v = \succ'$, and $\mathcal{S}_v = \emptyset$. 3. Add the edge $\langle u, v, f \rangle$ to $\mathcal{E}$. 4. Add $A' \to A'$ to $\mathcal{S}_v$ for each $A' \in K_2 \backslash \mathsf{core}'_v$, where 5. $\sigma = \{y \mapsto x, x \mapsto f(x)\}$ if $u \neq v_r$, or 6. $\sigma = \{y \mapsto o, x \mapsto f(o)\}$ if $u = v_r$, and 7. $K_1 = \{A' \in \mathsf{Su} \mid \top \to A'\sigma \in \mathcal{S}_u\}$, and 8. $K_2 = \{A' \in \mathsf{Su} \mid \Gamma' \to \Delta' \vee A' \in \mathcal{S}_u\}$ and $\Delta' \not\succeq_u A'\sigma$.

Table 2: Revised inference rules for the $\mathcal{ALCHIQ}$ calculus.

Join	If 1. $A \wedge \Gamma \to \Delta \in \mathcal{S}_v$, with A ground and o occurring in A, and 2. $\Gamma' \to \Delta' \vee \Delta'' \vee A \in \mathcal{S}_v$, with $\Delta' \cup \Delta'' \not\succeq_v A$, or 3. $\Gamma' \to \Delta' \vee A' \in \mathcal{S}_v$, with $\Delta' \not\succeq_v A'$, $A'\{x \mapsto o\} = A$, and $\Gamma' \to \Delta'' \vee x \approx o \in \mathcal{S}_v$, $\Delta'' \not\succeq_v x \approx o$, $\Gamma' = \top$, then add $\Gamma \wedge \Gamma' \to \Delta \vee \Delta' \vee \Delta''$ to $\mathcal{S}_v$.
r-Succ	If 1. $\Gamma \to \Delta \vee A\sigma \in \mathcal{S}_u$ where $\Delta \not\succeq_u A\sigma$, with $u \neq v_r$, 2. $A \in \mathsf{Su}^r$, with $o \in \Sigma_o^{\mathcal{O}}$ occurring in A, $\sigma = \{y \mapsto x\}$, and 3. there is no $\langle u, v_r, o \rangle \in \mathcal{E}$ s.t. $A \to A \,\hat{\in}\, \mathcal{S}_{v_r}$, and 4. (*) there is no $\Gamma'' \to \Delta'' \vee \bigvee_{i=1}^{n} L_i \in \mathcal{S}_u$ with $\Gamma'' \subseteq \Gamma$, $\Delta'' \subseteq \Delta$, and L_i of the form $x \approx o_i, y \approx o_i, x \approx y$, then add the edge $\langle u, v_r, o \rangle$ to $\mathcal{E}$ and $A \to A$ to $\mathcal{S}_{v_r}$.
r-Pred	If 1. $\bigwedge_{i=1}^{n} A_i \wedge \bigwedge_{i=1}^{m} C_i \to \bigvee_{i=1}^{k} L_i \in \mathcal{S}_{v_r}$, where $L_i \in \mathsf{Pr}^r$ for each nonground L_i, each C_i is ground, $A_i \in \mathsf{Su}^r$, and o_i is the named individual in A_i; and 2. there is $\langle u, v_r, o_i \rangle \in \mathcal{E}$ for each o_i such that $\Gamma_i \to \Delta_i \vee A_i\sigma \in \mathcal{S}_u$ verifies (*), $\Delta_i \not\succeq_u A_i\sigma$, $\sigma(y) = x$, then add $\bigwedge_{i=1}^{n} \Gamma_i \wedge \bigwedge_{i=1}^{m} C_i \to \bigvee_{i=1}^{n} \Delta_i \vee \bigvee_{i=1}^{k} L_i\sigma$ to $\mathcal{S}_u$.
Nom	If 1. $\bigwedge_{i=1}^{n} A_i \to \bigvee_{i=1}^{m} L_i \vee \bigvee_{i=m+1}^{k} L_i \in \mathcal{O}$, with L_i a-equalities 2. $\Gamma_i \to \Delta_i \vee A_i\sigma \in \mathcal{S}_{v_r}$, with $\Delta_i \not\succeq_{v_r} A_i\sigma$ and $\sigma(x) = o$, and 3. $L_i\sigma$ is of the form $y \approx y$ or $y \approx f_i(o_i)$ iff $m+1 \leq i \leq k$, then add $\Gamma \to \Delta \vee \bigvee_{i=1}^{K} y \approx o'_{\rho \cdot S^i}$ to $\mathcal{S}_{v_r}$, where $\Gamma = \bigwedge_{i=1}^{n} \Gamma_i$, $\Delta = \bigvee_{i=1}^{n} \Delta_i \vee \bigvee_{i=1}^{m} L_i\sigma$, and $K{+}1 = \max(i \mid \{z_i \text{ in } \mathcal{O}\})$

Table 3: Novel rules for reasoning with nominals in $\mathcal{ALCHOIQ}$.

to take into account that certain rules can be applied to the distinguished root context and that clauses propagated to predecessor contexts may contain ground atoms. The Core rule ensures that all atoms in a context's core hold. The Hyper rule performs hyperresolution between clauses in a context and ontology clauses; in prior work, variable x had to map to itself in the σ used in the rule, but now x maps to an individual if the rule is applied on the root context. The Eq, Ineq and Fact rules implement equality reasoning, and the Elim rule performs redundancy elimination as in prior work. The Pred rule performs hyperresolution between a context and a predecessor context. The rule does not apply to the root context, but the calculus provides another rule for that. Ground and nonground body atoms are treated differently; the latter are simply copied to the body of the derived clause. Finally, the Succ rule extends the context structure using the expansion strategy as in prior work. As in the Hyper rule, variable x maps to a named individual on the root context.

The rules in Table 3 handle reasoning with nominals. Rule Join corresponds to a resolution step between two ground atoms of different clauses in the same context. Rule r-Succ complements Succ by dealing with information propagation from any non-root context to the root context; in turn, r-Pred complements Pred in an analogous way. In contrast to previous calculi, rules r-Succ and r-Pred can be used to exchange information between the root context and any other context, not just a neighboring one; this is due to the fact that nominal reasoning is intrinsically non-local. Finally, rule Nom introduces additional nominals when an anonymous element of the canonical model may become arbitrarily interconnected. The Nom rule does not apply if the input ontology lacks either inverse roles, or nominals, or number restrictions.

3.2 The Reasoning Algorithm and its Properties

We can obtain a sound and complete reasoning algorithm for $\mathcal{ALCHOIQ}$ by exhaustively applying the inference rules in Tables 2 and 3 on a suitably initialised context structure. This follows from the calculus satisfying two properties analogous to those required by CB calculi in prior work. The *soundness property* ensures that the application of an inference rule to a sound context structure yields another sound context structure. The *completeness property* ensures that any query clause entailed by $\mathcal{O}$ will be contained up to redundancy in a suitably initialised context of a saturated context structure.

Theorem 1 (Soundness)**.** *Given a context structure $\mathcal{D}$ which is sound for $\mathcal{O}$, and an arbitrary expansion strategy, the application of a rule from Table 2 or Table 3 to $\mathcal{D}$ with respect to $\mathcal{O}$ yields a context structure which is sound for $\mathcal{O}$.*

Theorem 2 (Completeness)**.** *Let $\mathcal{D}$ be a context structure which is sound for $\mathcal{O}$ and such that no rule of Table 2 or Table 3 can be applied to it. Then, for each query clause $\Gamma_Q \to \Delta_Q$ and each context $q \in \mathcal{V}$ such that all of the following conditions hold, we have that $\Gamma_Q \to \Delta_Q \,\hat{\in}\, \mathcal{S}_q$ also holds.*

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

C2. For each context atom $A \in \Delta_Q$ and each A' of the form $B(x)$ such that $A \succ_q A'$, we have $A' \in \Delta_Q$.

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

To test whether $\mathcal{O}$ entails a query clause $\Gamma_Q \to \Delta_Q$, an algorithm can proceed as follows. In *Step 1*, create an empty context structure $\mathcal{D}$, and fix an expansion strategy. In *Step 2*, introduce a context q into $\mathcal{D}$, set its core to Γ_Q, and initialise the order $\succ_q$ in a way that is consistent with Condition C2 in Theorem 2. Finally, in *Step 3*, saturate $\mathcal{D}$ over the inference rules of the calculus and check whether $\Gamma_Q \to \Delta_Q$ is contained up to redundancy in $\mathcal{S}_q$. Such algorithm generalises to check in a single run a set of input query clauses by initialising in Step 2 a context q for each query clause.

Our algorithm may not terminate if the expansion strategy introduces infinitely many contexts. Termination is, however, ensured for strategies introducing finitely many contexts, such as those discussed in section 3.1.

Proposition 1. *The algorithm consisting of Steps 1–3 terminates if the expansion strategy introduces finitely many contexts and rule* Join *is applied eagerly. If the expansion strategy introduces at most exponentially many contexts, the algorithm runs in triple exponential time in the size of $\mathcal{O}$.*

Our algorithm is not worst-case optimal for $\mathcal{ALCHOIQ}$ (an NEXPTIME-complete logic), as it can generate a doubly exponential number of additional nominals and a number of clauses per context that is exponential in the size of the relevant signature; thus, each context can contain a triple exponential number of clauses in the size of $\mathcal{O}$. We can show, however, worst-case optimality for the well-known fragments of $\mathcal{ALCHOIQ}$, and thus obtain pay-as-you-go behaviour.

Proposition 2. *For any expansion strategy introducing at most exponentially many contexts, the algorithm consisting of Steps 1–3 runs in worst-case exponential time in the size of $\mathcal{O}$ if $\mathcal{O}$ is either $\mathcal{ALCHIQ}$, or $\mathcal{ALCHOQ}$, or $\mathcal{ALCHOI}$, or if it is Horn. Furthermore, for $\mathcal{ELHO}$ ontologies, the algorithm runs in polynomial time in the size of $\mathcal{O}$ with either the cautious or the eager strategy.*

Note that the strategies discussed in section 3.1 introduce at most exponentially many contexts. We conclude this section with an example illustrating the application of our calculus.

Example 3. Let $\mathcal{O}_1$ contain the following clauses.

$$A(x) \to R(x, f(x)) \quad (1) \qquad A(x) \to B_1(f(x)) \quad (2)$$
$$A(x) \to R(x, g(x)) \quad (3) \qquad A(x) \to B_2(g(x)) \quad (4)$$
$$B_1(x) \to S(o, x) \quad (5) \qquad B_2(x) \to S(o, x) \quad (6)$$
$$S(x, z_1) \wedge S(x, z_2) \to z_1 \approx z_2 \quad (7)$$
$$R(z_1, x) \wedge B_1(x) \wedge B_2(x) \to C(z_1) \quad (8)$$

We check whether $\mathcal{O}_1 \models A(x) \to C(x)$ using the eager expansion strategy. Figure 1 summarises the inferences relevant to deriving the query clause. Clauses 9 to 19 are attached to context v_A having core $A(x)$, clauses 20 to 23 to v_{B_1} with core $B_1(x) \wedge S(y, x)$, clauses 24 to 27 to v_{B_2} with core $B_2(x) \wedge S(y, x)$, and the remaining clauses to the root context v_r with empty core.

Consider the state of the context structure once clauses 9 to 13 and 20 to 22 have been derived as in the calculus from Bate *et al.* [2016]. We apply r-Succ to clause 22, which creates an o-labelled edge to v_r and adds clause 28; rule Nom derives clause 29 in v_r, with $o' = o_{S^1}$ an additional nominal. This clause can be back-propagated with r-Pred to yield

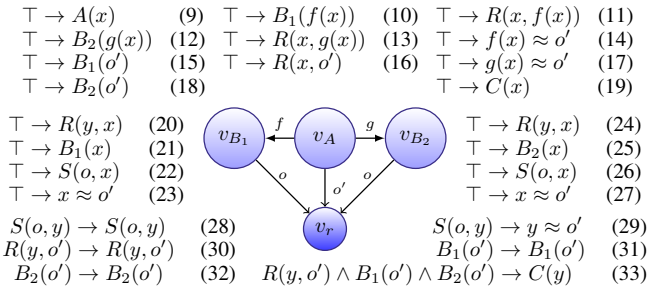

Figure 1: Calculus execution for Example 3. o' stands for o_{S^1}.

clause 23 in v_{B_1}; in turn, this can be back-propagated with Pred to yield clause 14 in v_A. Two applications of Eq yield clauses 15 and 16. Proceeding analogously in context v_{B_2}, we derive clauses 24 to 27 and then clauses 17 and 18. Next, we apply r-Succ to clauses 15, 16 and 18 to derive clauses 30 to 32. The Hyper rule can be applied to these clauses to derive clause 33. The head of this clause is in Pr^r, so we can back-propagate using r-Pred with the edge from v_A and clauses 15, 16 and 18 to derive our target, clause 19. ◁

4 Conclusion and Future Work

We have presented the first CB reasoning algorithm for a DL featuring all Boolean operators, role hierarchies, inverse roles, nominals, and number restrictions. We see many challenges for future work. First, our algorithm runs in triple exponential time, when it should be possible to devise a doubly exponential time algorithm; we believe, however, that deriving such tighter upper bound would require a significant modification of our approach. Second, our algorithm should be extended with datatypes in order to cover all of OWL 2 DL. Finally, we are implementing our algorithm as an extension of the Sequoia system [Bate *et al.*, 2016]. We expect good performance from the resulting system, as our calculus only steps beyond pay-as-you-go behaviour in the rare situation where disjunctions, nominals, number restrictions and inverse roles interact simultaneously.

Acknowledgments

Research supported by the SIRIUS centre for Scalable Data Access, and the EPSRC projects DBOnto, MaSI^3, and ED^3.

References

[Baader and Sattler, 2001] Franz Baader and Uli Sattler. An Overview of Tableau Algorithms for Description Logics. *Studia Logica*, 69:5–40, 2001.

[Baader *et al.*, 2003] Franz Baader, Diego Calvanese, Deborah L. McGuinness, Daniele Nardi, and Peter F. Patel-Schneider. *The Description Logic Handbook*. CUP, 2003.

[Baader *et al.*, 2005] Franz Baader, Sebastian Brandt, and Carsten Lutz. Pushing the $\mathcal{EL}$ envelope. In *IJCAI*, pages 364–369, 2005.

[Bachmair and Ganzinger, 2001] Leo Bachmair and Harald Ganzinger. Resolution Theorem Proving. In *Handbook of Automated Reasoning*, pages 19–99. Elsevier, 2001.

[Bate *et al.*, 2016] Andrew Bate, Boris Motik, Bernardo Cuenca Grau, Franstisek Simancik, and Ian Horrocks. Extending consequence-based reasoning to SRIQ. In *KR*. pp 187-96, AAAI, 2016.

[Ganzinger and De Nivelle, 1999] Harald Ganzinger and Hans De Nivelle. A Superposition Decision Procedure for the Guarded Fragment with Equality. In *LICS*, pages 295–305, Trento, Italy, July 2–5 1999. IEEE Computer Society.

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

[Glimm *et al.*, 2014] Birte Glimm, Ian Horrocks, Boris Motik, Giorgos Stoilos, and Zhe Wang. HermiT: An OWL 2 Reasoner. *J. of Automated Reasoning*, 53(3):245–269, 2014.

[Haarslev *et al.*, 2012] Volker Haarslev, Kay Hidde, Ralf Möller, and Michael Wessel. The RacerPro knowledge representation and reasoning system. *Semantic Web*, pages 267–277, 2012.

[Horrocks and Sattler, 2005] Ian Horrocks and Ulrike Sattler. A tableaux decision procedure for $\mathcal{SHOIQ}$. In *IJCAI*, pages 448–453, 2005.

[Horrocks *et al.*, 2006] Ian Horrocks, Oliver Kutz, and Uli Sattler. The even more irresistible $\mathcal{SROIQ}$. *KR*, pp.57–67, 2006.

[Hustadt and Schmidt, 1999] Ullrich Hustadt and Renate A. Schmidt. Issues of Decidability for Description Logics in the Framework of Resolution. In *Selected Papers from Automated Deduction in Classical and Non-Classical Logics*. pp.191–205, 1999.

[Hustadt and Schmidt, 2002] Ullrich Hustadt and Renate A. Schmidt. Using Resolution for Testing Modal Satisfiability and Building Models. *J. of Automated Reasoning*, 28(2):205–232, 2002.

[Hustadt *et al.*, 2004] Ullrich Hustadt, Boris Motik, and Ulrike Sattler. Reducing $\mathcal{SHIQ}^-$ description logic to disjunctive datalog programs. In *KR*, pages 152–162, 2004.

[Hustadt *et al.*, 2008] Ullrich Hustadt, Boris Motik, and Ulrike Sattler. Deciding Expressive Description Logics in the Framework of Resolution. *Information & Computation*, 206(5):579–601, 2008.

[Karahroodi and Haarslev, 2017] Nikoo Z. Karahroodi and Volker Haarslev. A consequence-based algebraic calculus for SHOQ. In *DL*, 2017.

[Kazakov and Motik, 2006] Yevgeny Kazakov and Boris Motik. A resolution-based decision procedure for $\mathcal{SHOIQ}$. In *IJCAR*, pages 662–677. Springer, 2006.

[Kazakov *et al.*, 2012] Yevgeny Kazakov, Markus Krötzsch, and Frantivsek Simanvcík. Practical Reasoning with Nominals in the EL Family of Description Logics. *KR*, 2012.

[Kazakov, 2009] Yevgeny Kazakov. Consequence-driven reasoning for Horn SHIQ ontologies. In *IJCAI*. pp 2040–45, 2009.

[Motik *et al.*, 2007] Boris Motik, Rob Shearer, and Ian Horrocks. Optimized reasoning in description logics using hypertableaux. In *CADE*, pages 67–83. Springer, 2007.

[Motik *et al.*, 2009] Boris Motik, Rob Shearer, and Ian Horrocks. Hypertableau reasoning for description logics. *JAIR*, 36:165–228, 2009.

[Nieuwenhuis and Rubio, 2001] Robert Nieuwenhuis and Albert Rubio. Paramodulation-Based Theorem Proving. In *Handbook of Automated Reasoning*. pp.371–443.Elsevier, 2001.

[Nivelle *et al.*, 2000] 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.

[Ortiz *et al.*, 2010] Magdalena Ortiz, Sebastian Rudolph, and Mantas Simkus. Worst-Case Optimal Reasoning for the Horn-DL Fragments of OWL 1 and 2. *KR*, 2010.

[Schmidt and Hustadt, 2007] Renate A. Schmidt and Ullrich Hustadt. The Axiomatic Translation Principle for Modal Logic. *ACM Transactions on Comp. Logic*, 8(4), 2007.

[Schmidt and Hustadt, 2013] Renate A. Schmidt and Ullrich Hustadt. First-Order Resolution Methods for Modal Logics. In *Programming Logics—Essays in Memory of Harald Ganzinger*, pages 345–391. Springer, 2013.

[Simančík *et al.*, 2011] František Simančík, Yevgeny Kazakov, and Ian Horrocks. Consequence-based reasoning beyond Horn ontologies. In *IJCAI*, pages 1093–1098, 2011.

[Simančík *et al.*, 2014] František Simančík, Boris Motik, and Ian Horrocks. Consequence-based and fixed-parameter tractable reasoning in description logics. *Artificial Intelligence*, 209:29–77, 2014.

[Sirin *et al.*, 2007] Evren Sirin, Bijan Parsia, Bernardo Cuenca Grau, Aditya Kalyanpur, and Yarden Katz. Pellet: A practical OWL-DL reasoner. *Journal of Web Semantics*, 5(2):51–53, 2007.

[Steigmiller *et al.*, 2014] Andreas Steigmiller, Thorsten Liebig, and Birte Glimm. Konclude: System description. *J. of Web Semantics*, 27(1), 2014.

[Tena Cucala *et al.*, 2017] David Tena Cucala, Bernardo Cuenca Grau, and Ian Horrocks. Consequence-based reasoning for description logics with disjunction, inverse roles and nominals. *DL*, 2017.

[Tena Cucala *et al.*, 2018] David Tena Cucala, Bernardo Cuenca Grau, and Ian Horrocks. Consequence-based reasoning for description logics with disjunction, inverse roles, number restrictions, and nominals. arxiv.org/abs/1805.01396, 2018.

[Tsarkov and Horrocks, 2006] Dmitry Tsarkov and Ian Horrocks. FaCT++ Description Logic Reasoner: System Description. In *IJCAR*, pages 292–297. Springer, 2006.