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

David Tena Cucala, Bernardo Cuenca Grau, and Ian Horrocks

Department of Computer Science, University of Oxford, UK
first.last[.2nd last]@cs.ox.ac.uk

Abstract. We present a consequence-based calculus for the $\mathcal{SHOI}$ description logic. Known consequence-based reasoning procedures are either restricted to logics without nominals (e.g. $\mathcal{SRIQ}$), or to Horn logics (e.g. Horn-$\mathcal{SROIQ}$). In line with existing approaches, our algorithm uses context structures to assist in the derivation of clauses, which are generated by inference rules such as (hyper)resolution. These rules are applied only locally—that is, between clauses in either the same context or in neighbouring contexts. In order to deal with interactions between nominals, inverse roles, and disjunction, we allow for a restricted form of non-local inference. Our calculus displays worst-case optimal behaviour, and it is amenable to implementation.

1 Introduction

Description logics (DLs) [2] are a family of languages for knowledge representation. They underpin the Semantic Web standard OWL 2 [5], and are used for conceptual modelling across a wide spectrum of application domains, ranging from medicine [22] to oil and gas exploration [12]. An important benefit of DL-based languages is the use of reasoning to support modelling, with concept subsumption and classification being key reasoning services.

Consequence-based (CB) reasoning [9] has emerged as a successful paradigm for efficiently solving such reasoning tasks. This approach combines some of the most effective features of (hyper)tableau [7, 21, 25, 23, 15] and resolution-based [8, 14, 11] reasoning algorithms. On the one hand, similarly to resolution, CB algorithms process DL ontologies by deriving entailed formulae and hence avoiding the explicit construction of potentially large models. On the other hand, CB algorithms are guided by a graph structure similar to a (hyper)tableau, which prevents them from drawing many unnecessary inferences and, as a result, leads to a nice goal-oriented behaviour. Finally, CB techniques are particularly suitable for classification tasks, since they can generally verify a large number of subsumptions in a single execution, allowing for "one-pass" full classification.

CB reasoning algorithms were originally proposed for the $\mathcal{EL}$ family of lightweight description logics [1, 10], and later extended to the more expressive logics Horn-$\mathcal{SHIQ}$ [9], Horn-$\mathcal{SROIQ}$ [16], and $\mathcal{ALCH}$ [19]. A unifying framework for

consequence-based reasoning was developed in [20], introducing the notion of *contexts* as a mechanism for constraining resolution inferences and making them goal-directed. The framework was extended to other non-Horn DLs such as $\mathcal{SHI}$ [20] and $\mathcal{SRIQ}$ [4].

We continue this line of work by exploring the extension of CB techniques to non-Horn DLs with nominals. The combination of nominals, inverse roles, and disjunctions poses new challenges, such as the need for deriving clauses expressing non-local dependencies between contexts. We show how to extend the CB calculus for $\mathcal{SHI}$ [19] in order to overcome these issues, and we prove that the resulting calculus is a worst-case optimal decision procedure for subsumption in $\mathcal{SHOI}$. We also show that the algorithm displays pay-as-you-go behaviour and, in particular, works in polynomial time when restricted to $\mathcal{ELHO}$ ontologies.

Our proposal is related to the deterministic ExpTime tableau calculi for $\mathcal{SHOI}$ presented in [13]. This algorithm uses global caching, which allows the re-use of nodes in cases where two distinct individuals would have the same label. In comparison, our algorithm can accommodate more intensive reuse policies, because nodes represent groups of individuals that can have very different types. We also handle disjunction by means of hyper-resolution on non-Horn clauses (as in other CB algorithms), rather than by or-branching.

2 Preliminaries

First-Order Logic We use standard first-order logic terminology. A signature Σ is pair of finite sets (Σ^A, Σ^f) where Σ^A is set of *predicates* and Σ^f is a set of of *function symbols* of arity ≥ 0. A *clause* is a formula of the form $\forall \boldsymbol{x}.\Gamma \rightarrow \Delta$, where Γ (resp. Δ), called the *body* (resp. *head*) of the clause, is a possibly empty conjunction (resp. disjunction) of atoms, and $\forall \boldsymbol{x}$ is a universal quantification over all variables occurring in Γ and Δ, which is often omitted. We use $\top$ and $\bot$ to represent the empty conjunction and disjunction, respectively. We often use set notation to refer to relations between atoms, conjunctions, disjunctions, and clauses.

A substitution σ is a mapping from variables to terms. It can be expressed as $\{x_1 \mapsto t_1, x_2 \mapsto t_2, \dots\}$, which contains all mappings in σ different from the identity. The result of applying σ to a clause C or term t is written $C\sigma$ and $t\sigma$, respectively. A position p in a term $t = f(t_1, \dots, t_n)$ is a sequence of natural numbers $i_1 \circ i_2 \circ \cdots \circ i_k$ which identifies a particular subterm of t. Positions are defined inductively, with $t|_\epsilon = t$ and $t|_p = t_i|_{p'}$ for any p, p', i such that $p = i \circ p'$. If $t|_p = s$, we write $t[s']_p$ for the result of replacing the subterm s by s' in position p of term t. Positions are defined analogously for atoms.

An interpretation I is a pair $(D^I, \cdot^I)$, where D^I is a non-empty set (the *domain*), and $\cdot^I$ is a function which maps every $f \in \Sigma^f$ of arity $k \in \mathbb{N}$ to a function from $(D^I)^k$ to D^I, and every predicate symbol of arity $k \in \mathbb{N}$ to a subset of $(D^I)^k$. For each first-order sentence φ, an interpretation I induces a truth-value φ^I in the usual way. An interpretation I satisfies a sentence φ iff $\varphi^I = \mathsf{true}$, in which case we write $I \models \varphi$.

Orders A *(strict) order* $\succ$ on a non-empty set S is a binary, irreflexive, transitive, and antisymmetric relation between elements of S. An order $\succ$ induces a *non-strict* order $\succeq$ if it is extended with reflexivity. An order is *total* if for any distinct $a, b \in S$, either $a \succ b$ or $b \succ a$. For any $\circ \in \{\succ, \succeq\}$, we write $S \circ a$ if there is $b \in S$ with $b \circ a$. An order on a set S can be extended to an order $\succ'$ between subsets of S: given two subsets S_1, S_2 of S and an order $\succ$ on S, we have that $S_1 \succ' S_2$ if and only if for each element $n \in S_2 \backslash S_1$ there is an element $m \in S_1 \backslash S_2$ such that $m \succ n$. Since the meaning will be clear from the context, we refer to an induced order $\succ'$ also as $\succ$.

The $\mathcal{SHOI}$ description logic A $\mathcal{SHOI}$ ontology in the usual DL syntax can be transformed in polynomial time and expressed as an equisatisfiable set of first-order clauses as indicated in Table 1, where the left-hand side contains each possible normalised DL axiom form, and the right-hand side contains the corresponding first-order logic clause[1]. Transitivity axioms are encoded away through the usual techniques [18, 6]; other details of the normalisation are well-known [9] and it is straightforward to extend the procedure to include nominals.

Clauses with nominals in first-order logic can be represented using the equality predicate $\approx$. In order to deal with this predicate, reasoning procedures either exploit paramodulation techniques [26], or explicitly axiomatise equality. However, the inference rules of our algorithm have equality reasoning embedded into them. This allows us to choose a first-order representation formalism with the property that the equality predicate will never appear in the clauses manipulated by the algorithm. In order to achieve this, we define a function π mapping each nominal o in the ontology to a fresh unary predicate $\pi(o) = O$, and use this predicate in the translation of each DL axiom or fact which features o. If Σ^o is the set of all nominals in the ontology signature, we denote as Σ^O the set $\{\pi(o) \mid o \in \Sigma^o\}$. In order to preserve satisfiability, for each $O \in \Sigma^O$, we add an axiom of the form DL8, and we assume that the equality predicate is explicitly axiomatised. *However*, these axioms will be ignored by our algorithm, as the inference rules only use axioms of the form DL1-DL7, thereby preventing equality from appearing in derived clauses; this ontology subset is denoted $\mathcal{O}^*$.

Table 1. Normalised $\mathcal{SHOI}$ axioms and their translation to clauses.

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	$B_1 \sqsubseteq \exists R.B_2$	$\rightsquigarrow$	$B_1(x) \rightarrow R(x, f(x))$ $B_1(x) \rightarrow B_2(f(x))$
DL3	$\exists R.B_1 \sqsubseteq B_2$	$\rightsquigarrow$	$R(z, x) \wedge B_1(x) \rightarrow B_2(z)$
DL4	$B \sqsubseteq \{o\}$	$\rightsquigarrow$	$B(x) \rightarrow O(x)$
DL5	$B(o)$	$\rightsquigarrow$	$O(x) \rightarrow B(x)$
DL6	$R_1 \sqsubseteq R_2$	$\rightsquigarrow$	$R_1(z, x) \rightarrow R_2(z, x)$
DL7	$R_1 \sqsubseteq R_2^-$	$\rightsquigarrow$	$R_1(z, x) \rightarrow R_2^-(x, z)$
DL8	$-$		$O(x) \rightarrow x \approx o$

[1] The type of the generated clauses can be seen as a subset of a broader kind, named *DL-clauses*, which are generally used to express axioms of DLs in first-order logic.

3 The Consequence-based Framework

Our proposal builds upon the CB algorithms in [20] and [4]. Even though the original framework in [20] used exclusively DL notation to represent both input and derived clauses, we use the first-order clause notation introduced in the extension to $\mathcal{SRIQ}$ [4], as DL notation proved to be insufficiently expressive to represent all types of relevant derived consequences. This choice also becomes necessary when we combine inverse roles with nominals, and it establishes a common framework between both algorithms.

The CB algorithms introduced above take an ontology as an input and use it to construct a *context structure*—a graph where each node v (a *context*), represents a unique, specific set of individuals, and contains clauses that hold *only* for those individuals, while each edge is labelled by a function symbol $f \in \Sigma^f$ which is used to represent f-successor relations between terms.

The set of individuals represented by each v is determined by a *core*, which is a collection of atoms core_v in variables y and x with no function symbols. Each v represents those pairs of individuals (t_1, t_2) in a model I such that I satisfies the grounding of the core by $\sigma = \{x \mapsto t_1, y \mapsto t_2\}$. *However*, context structures are constrained to ensure that any such pair of individuals is of the form $(f^I(s^I), s^I)$ for some term s and $f \in \Sigma^f$ in the label of an incoming edge. Therefore, we should see each context v as a representation of those individuals t which: (i) are instances of every $B(x) \in \mathsf{core}_v$ and (ii) if $t = f(t')$ for some f in an incoming edge from u, then t' is represented in u and is an R-successor (or R-antecessor) of t for each $R(x, y)$ (or $R(y, x)$) in core_v. Consequently, for the clauses in a context, variable x ranges over each of these elements t, and variable y represents its immediate predecessor. This stands in contrast with the meaning of variables x and z in Table 1, which range over the interpretation domain as usual.

A set $\mathcal{S}_v$ is defined for each v, and it contains clauses holding specifically for those individuals represented by v; these clauses should only use variables y and x. A key feature of the consequence-based framework is that only functional terms of the form $f(x)$ (with $f \in \Sigma^f$), are allowed in each $\mathcal{S}_v$. This restriction simplifies the execution of the algorithm, and prevents term nesting, which can lead to termination problems. Moreover, this implies that clauses derived by the algorithm describe constraints which are strictly *local*, that is, which involve exclusively an element t in the interpretation domain, its possible successors $f_1(t), \cdots, f_n(t)$, and its ancestor t', if it exists. However, this does not mean they are *localised*, since each clause holds not only for one element in the domain, but rather for *any* element represented by the context.

Once a context structure has been initialised, a set of inference rules is used to expand it by creating new contexts, edges, and deriving new clauses within contexts. These rules also act locally: they only take as input sets of clauses in a particular context v and (possibly) its neighbour, and add their output to these contexts, or to a newly created neighbour of v.

However, when nominals are allowed in the ontology, some form of non-local interaction is required, since individuals belonging to unrelated contexts

(including contexts that are not even connected in the context structure) can collapse into each other, which may give rise to new consequences that need to be accounted for. As we explain in the next section, our algorithm deals with this issue by allowing clauses to express some non-local restrictions, and by introducing non-local inference rules.

4 Combining Nominals, Inverse Roles, and Disjunction

In this section, we demonstrate how the CB approach introduced in [19], with support for disjunction and inverse roles, can be adapted to deal with nominals. We illustrate this by applying our calculus to determine that $A \sqsubseteq F$ is entailed by the ontology $\mathcal{O}$ given in Table 2, which uses nominals in a non-trivial way.

Suppose we attempt to build a Herbrand model using (a variant of) the hypertableau method. We start the procedure with an element c and the atom $A(c)$. By axioms (1)-(6), we produce descendants $f_3(f_1(c))$ and $f_2(c)$, in the atoms $B_1(f_3(f_1(c)))$ and $B_2(f_2(c))$. By axioms (12) and (18), we also have $B_3(o_3)$. Now, axioms (13)-(18), imply that each of the elements $f_3(f_1(c)), f_2(c)$, and o_3 must be equal to either o_1 or o_2 (or both). Therefore, at least two of these three elements must be equal, *even though we do not know which nominal they are equal to.* As a result, in any branch, one of the axioms (7)-(9) can be used to derive either $C(f_2(c))$ or $C(f_3(f_1(c)))$. This can resolved with axioms (10) and/or (11) to obtain $F(c)$, as desired.

The restriction that two of the three elements in $\{f_3(f_1(c)), f_2(c), o_3\}$ must be equal is no longer local; in fact, $f_3(f_1(c))$ and $f_2(c)$ are not even connected to o_3 in the tableau. This suggests that a consequence-based procedure should contain clauses which represent relations other than those between neighbouring individuals. However, if we allow for more variables in a context, to refer to elements in arbitrary distant contexts, the consequence-based method quickly collapses into a procedure analogous to resolution.

Instead, our algorithm allows only references to nominals in context clauses, in addition to the local individuals represented by $y, x, f(x)$. This introduces the possibility of having ground atoms in context clauses. We also introduce non-local inference rules, which take a context as an input, and either a (possibly disconnected) *nominal context*, which represents a single nominal, or a special context known as the *root* context, where ground resolution can be applied.

In Figure 1, we illustrate how the algorithm works by applying it to the ontology $\mathcal{O}$ of Table 2. The algorithm is initialised with a *root* context v_0 with core $A(x)$, corresponding to the body of our query. We also create contexts v_{o_1}, v_{o_2}, and v_{o_3} for the corresponding nominals, with cores $O_1(x)$, $O_2(x)$, and $O_3(x)$, respectively. Back in the root context, we use hyperresolution with axioms (1)-(4) to derive clauses (23)-(26). Observe that we cannot resolve clause (23) with axioms (5) or (6), since that would lead to nested function terms. Instead, we first have to create a successor context v_1 with a Succ rule, apply the Core rule to generate the equivalent clause (28), and then we can apply those axioms to generate clause (30). Further applications of hyperresolution and generation of

Table 2. Ontology $\mathcal{O}$

$A \sqsubseteq \exists R.D$	$\rightsquigarrow$	$A(x) \rightarrow D(f_1(x))$	(1)
		$A(x) \rightarrow R(x, f_1(x))$	(2)
$A \sqsubseteq \exists R.B_2$	$\rightsquigarrow$	$A(x) \rightarrow B_2(f_2(x))$	(3)
		$A(x) \rightarrow R(x, f_2(x))$	(4)
$D \sqsubseteq \exists R.B_1$	$\rightsquigarrow$	$D(x) \rightarrow B_1(f_3(x))$	(5)
		$D(x) \rightarrow R(x, f_3(x))$	(6)
$B_1 \sqcap B_2 \sqsubseteq C$	$\rightsquigarrow$	$B_1(x) \wedge B_2(x) \rightarrow C(x)$	(7)
$B_1 \sqcap B_3 \sqsubseteq C$	$\rightsquigarrow$	$B_1(x) \wedge B_3(x) \rightarrow C(x)$	(8)
$B_2 \sqcap B_3 \sqsubseteq C$	$\rightsquigarrow$	$B_1(x) \wedge B_3(x) \rightarrow C(x)$	(9)
$\exists R.C \sqsubseteq F$	$\rightsquigarrow$	$R(z, x) \wedge C(x) \rightarrow F(z)$	(10)
$\exists R.F \sqsubseteq F$	$\rightsquigarrow$	$R(z, x) \wedge F(x) \rightarrow F(z)$	(11)
$\{o_3\} \sqsubseteq B_3$	$\rightsquigarrow$	$O_3(x) \rightarrow B_3(x)$	(12)
$B_1 \sqsubseteq \{o_1\} \sqcup \{o_2\}$	$\rightsquigarrow$	$B_1(x) \rightarrow O_1(x) \vee O_2(x)$	(13)
$B_2 \sqsubseteq \{o_1\} \sqcup \{o_2\}$	$\rightsquigarrow$	$B_2(x) \rightarrow O_1(x) \vee O_2(x)$	(14)
$B_3 \sqsubseteq \{o_1\} \sqcup \{o_2\}$	$\rightsquigarrow$	$B_3(x) \rightarrow O_1(x) \vee O_2(x)$	(15)
		$O_1(x) \rightarrow x \approx o_1$	(16)
		$O_2(x) \rightarrow x \approx o_2$	(17)
		$O_3(x) \rightarrow x \approx o_3$	(18)

new successors lead to the clause $\top \rightarrow O_1(x) \vee O_2(x)$ appearing in contexts v_3 (which represents, among others, the term $f_3(f_1(c))$), v_2 (id. for $f_2(c)$) and v_4 (id. for o_3). We are now in the situation described above, where non-local inferences are required to derive further consequences.

We achieve this by using a new rule Split which generates atoms grounded in nominals. For instance, we apply this rule to clauses (33) and (35), selecting $O_2(x)$ to generate clause (42). The Split rule also introduces clause (43) in the context v_{o_2} corresponding to the selected nominal, in order to explore the consequences of $B_1(o_2)$. We follow the same procedure in v_2, which adds clause (48) also to v_{o_2}. These two clauses can now be resolved with axiom (9) to generate (49). This clause can be relevant to contexts v_3 and v_2, so we propagate it back to these contexts with a new rule Exp and obtain (50). Now $C(x)$ can be resolved together with clause (34) with axiom (10) to derive (51). Then, we can apply again Split, selecting $O_1(x)$ and, after a similar procedure, we derive clause (58). Since the head of this clause involves exclusively the predecessor of x, it can be propagated back to context v_1 with a rule Pred, and then, with axiom (11) and again the Pred rule, we derive (61) in v_0.

In order to derive $F(x)$ unconditionally in this context, we must show that the ontology either satisfies both $B_2(o_2)$ and $B_3(o_1)$, or it derives $F(x)$ by other means. We can explore the alternatives to satisfying $B_2(o_2)$ starting from clause (47). Using a similar procedure to the one presented above with Split and Exp rules, and propagating the result (66) to the root context with Pred, we obtain (67). We then use a rule Join to resolve ground atoms in the root context to derive (68). Now, since clause (54) is grounded, we can copy it in the root context using a rule Root specifically for this purpose. Using Join leads to (70).

The only remaining task is to follow a symmetric procedure to derive (83), and using Join once again to obtain $\top \to F(x)$ in the root context.

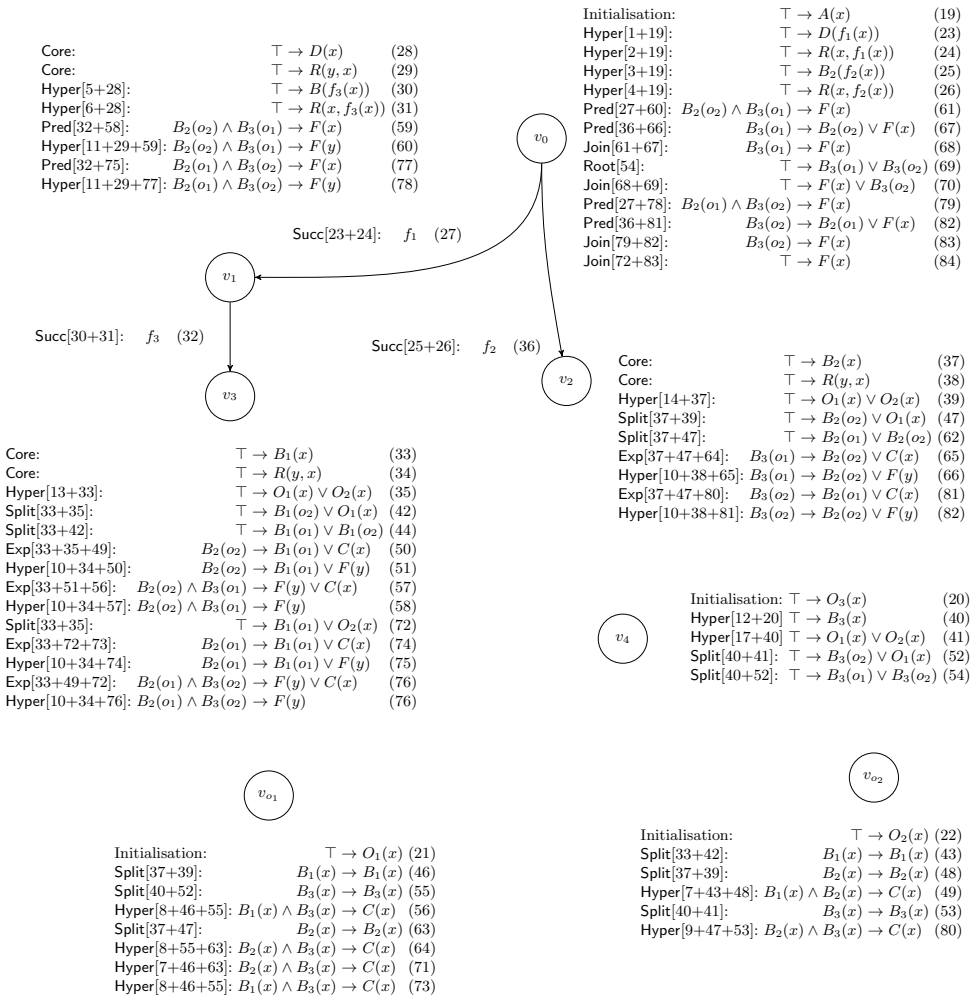

Fig. 1. Execution of the $\mathcal{SHOI}$ calculus on ontology $\mathcal{O}$

5 The $\mathcal{SHOI}$ Calculus

We introduce first some key definitions which extend those in [4].

Definition 1 (Context terms, literals, and clauses). *A context term is a term of the form x, y, $f(x)$ for some $f \in \Sigma^f$, or o for some $o \in \Sigma^o$. A context*

atom is an atom of the form $C(x)$, $C(y)$, $C(f(x))$, $R(x,y)$, $R(y,x)$, $R(x,f(x))$, $R(f(x),x)$, or $C(o)$, with R a binary predicate in Σ^A, and C a unary predicate in $\Sigma^A \cup \Sigma^O$. A context clause is a clause where all literals are context atoms, and which contains no function symbols in the body. A query clause contains only unary predicates $B \in \Sigma^A$.

The sets we define next, called *trigger* sets, are collections of atoms which guide the application of inference rules. To give an example of how they work, consider the Succ rule, which carries information from a context v to a successor. The rule can only be applied if there is a context with a clause such that a successor trigger is in the clause.

Definition 2 (Trigger sets). *The set of successor triggers is defined as follows*

$$\begin{aligned}\mathsf{Su}(\mathcal{O}) = \{B(x) \mid \textit{exists } \Gamma \to \Delta \in \mathcal{O} \textit{ s.t. } B(x) \in \Gamma \textit{ and } B \in \Sigma^A\} \cup \\ \{R(y,x) \mid \textit{exists } \Gamma \to \Delta \in \mathcal{O} \textit{ s.t. } R(z,x) \in \Gamma\} \cup \\ \{R(x,y) \mid \textit{exists } \Gamma \to \Delta \in \mathcal{O} \textit{ s.t. } R(x,z) \in \Gamma\}\end{aligned}$$

In turn, the set of predecessor triggers is:

$$\mathsf{Pr}(\mathcal{O}) = \{B(y) \mid B \in \Sigma^A\} \cup \{A\{x \mapsto y, y \mapsto x\} \mid A \in \mathsf{Su}(\mathcal{O})\} \cup \mathsf{Gr}(\mathcal{O})$$

where $\mathsf{Gr}(\mathcal{O}) = \{C(o) \mid C \in (\Sigma^A \cup \Sigma^O), o \in \Sigma^o\}$.

Observe that $\mathsf{Pr}(\mathcal{O})$ does not contain any $O(y)$, and $\mathsf{Su}(\mathcal{O})$ does not contain any $O(x)$. We also define an ordering for the atoms in the clauses; adding order to the technique reduces the amount of required computation.

Definition 3 (Context order). *Let $\succcurlyeq$ be an arbitrary total order on function symbols. Let $\succcurlyeq'$ be an order induced by $\succcurlyeq$ on terms such that $f(x) \succcurlyeq' x \succcurlyeq' y$, for any $f \in \Sigma^f$ and for each pair f, g of function symbols such that $f \succcurlyeq g$, we have $f(x) \succcurlyeq' g(x)$. A context order $\succ$ is a strict order on context atoms such that: (i) For each atom A, if $t \succcurlyeq' s$ then $A[t]_p \succ A[s]_p$, (ii) for any atom $A \in \mathsf{Pr}(\mathcal{O})$ with variable y, we have $A \not\succ A'$ for any other context atom A'.*

Such an order always exists: consider an arbitrary lexicographic path order [3] induced by $\succcurlyeq'$ on the atoms, which guarantees that condition (i) is verified. Then, relax the order to accommodate condition (ii) by making the atoms in $A' \in \mathsf{Pr}(\mathcal{O})$ smallest, including also the atoms in $\mathsf{Gr}(\mathcal{O})$. This does not violate condition (i), for no atom of the form $B(x)$ appears in $\mathsf{Pr}(\mathcal{O})$.

We use a notion of redundancy to simplify the calculus by allowing for elimination of clauses that are subsumed by others in the context structure.

Definition 4 (Redundancy). *A set of clauses U contains a clause $\Gamma \to \Delta$ up to redundancy (abbreviated u.t.r.), if and only if there exist $\Gamma' \subseteq \Gamma$ and $\Delta' \subseteq \Delta$ such that $\Gamma' \to \Delta' \in U$. We write $\Gamma \to \Delta \,\hat{\in}\, U$.*

The annotated graph which guides the algorithm execution is defined next:

Definition 5 (Context structure). *A context structure for an ontology $\mathcal{O}$ is a tuple $\mathcal{D} = \langle \mathcal{V}, \mathcal{E}, \mathsf{core}, \mathcal{S}, \succ, r \rangle$, where $\mathcal{V}$ is a finite set of elements denominated* contexts*; $\mathcal{E}$ is a subset of $\mathcal{V} \times \mathcal{V} \times \Sigma^f$, that is, a finite set of directed edges between contexts, labelled by a function symbol; core is a function that assigns to each context a conjunction core_v of atoms of $\mathsf{Su}(\mathcal{O})$; $\mathcal{S}$ is a function which maps every context v to a set of context clauses $\mathcal{S}_v$; $\succ$ is a function which assigns to each context v a context order $\succ_v$, and $r(\mathcal{V})$ is a function which chooses a context of $\mathcal{V}$, denominated the root context.*

The following defines a notion of soundness for context structures, which is preserved as an invariant during the algorithm execution.

Definition 6 (Sound context structure). *A context structure is sound if and only if for each context $v \in V$ and each context clause $\Gamma \to \Delta$ in $\mathcal{S}_v$, we have:*

$$\mathcal{O} \models \forall x \forall y. \, (\mathsf{core}_v(x, y) \wedge \Gamma \to \Delta) \,,$$

and for each edge $\langle u, v, f \rangle \in \mathcal{E}$, we have:

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

The rules of the algorithm require an *expansion strategy*, which is a "policy" for deciding whether to create new contexts or re-use existing ones.

Definition 7 (Expansion strategy). *An expansion strategy strat is a polynomially computable function which takes as input a triple $(f, K, \mathcal{D})$, and returns another triple $(v, \mathsf{core}, \succ)$. In the input, f is a function symbol, K is a set of atoms with predicates in Σ^A, and $\mathcal{D}$ is a context structure. The output consists of a core $\mathsf{core} \subseteq K$ which does not contain predicates in Σ^O, a context v which is either fresh or such that $\mathsf{core} = \mathsf{core}_v$, and a context order $\succ$.*

Typically, three main *natural* expansion strategies are considered, depending on which nodes and core functions they return (the context order is chosen arbitrarily): the *trivial* strategy, which always returns $(v_\top, \top)$, with $v_\top$ a fixed context with empty core; the *cautious* strategy, which returns $(v_B, B(x))$ if f occurs in $\mathcal{O}$ only in an atom $B(f(x))$, and $(v_\top, \top)$ otherwise, and the *eager* strategy, which returns (v_K, K); the latter was used in the example above.

The inference rules for our calculus are in Table 3. We assume, for simplicity, that repeated atoms are eliminated before adding a clause to a context. An execution of the algorithm on query $\Gamma_Q \to \Delta_Q$ chooses an expansion strategy and initialises a context structure $\mathcal{D}$ with a context $r(\mathcal{V}) = q$ with $\mathsf{core}_q = \Gamma_Q$, an order $\succ_q$ which satisfies condition C2 of Theorem 2, a context v_o for each $o \in \Sigma^o$, with $\mathsf{core}_{v_o} = O(x)$, and an order $\succ_o$. Then, it saturates $\mathcal{D}$ by applying the inference rules in table 3 w.r.t. $\mathcal{O}^*$ until no further rules can be applied. Due to the following claims, which we prove in [24], we then have that $\Gamma_Q \to \Delta_Q \,\hat{\in}\, \mathcal{S}_q$ if and only if the query is entailed by $\mathcal{O}$.

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

Table 3. Rules of the Consequence-Based Calculus for $\mathcal{SHOI}$

Core	If	$A \in \mathsf{core}_v$
	then	add $C = \top \to A$ to $\mathcal{S}_v$, unless $C \,\hat{\in}\, \mathcal{S}_v$ already.
Hyper	If	$\bigwedge_{i=1}^{n} A_i \to \Delta \in \mathcal{O}$, σ is such that $x \mapsto x$
		and $\Gamma_i \to \Delta_i \vee A_i\sigma \in \mathcal{S}_v$ with $\Delta_i \not\succeq_v A_i\sigma$, for $1 \le i \le n$
	then	add $C = \bigwedge_{i=1}^{n} \Gamma_i \to \Delta\sigma \vee \bigvee_{i=1}^{n} \Delta_i$ to $\mathcal{S}_v$, unless $C \,\hat{\in}\, \mathcal{S}_v$ already.
Elim	If	$\Gamma \to \Delta \in \mathcal{S}_v$
		and $\Gamma \to \Delta \,\hat{\in}\, \mathcal{S}_v \backslash (\Gamma \to \Delta)$
	then	remove $\Gamma \to \Delta$ from $\mathcal{S}_v$.
Pred	If	$\Gamma_O \wedge \bigwedge_{i=1}^{m} A_i \to \bigvee_{i=m+1}^{m+n} L_i \in \mathcal{S}_v$ with $L_i \in \mathsf{Pr}(\mathcal{O})$ for each
		$m+1 \le i \le m+n$, and $\Gamma_O \subseteq \mathsf{Gr}(\mathcal{O})$,
		and there is $\langle u, v, f\rangle \in \mathcal{E}$ such that
		$\Gamma_i \to \Delta_i \vee A_i\sigma \in \mathcal{S}_u$ where $\Delta_i \not\succeq_u A_i\sigma$, for $1 \le i \le m$.
		with $\sigma = \{y \mapsto x, x \mapsto f(x)\}$
	then	add $C = \Gamma_O \wedge \bigwedge_{i=1}^{m} \Gamma_i \to \bigvee_{i=1}^{m} \Delta_i \vee \bigvee_{i=m+1}^{m+n} L_i\sigma$ to $\mathcal{S}_u$,
		unless $C \,\hat{\in}\, \mathcal{S}_u$ already.
Succ	If	$\Gamma \to \Delta \vee A \in \mathcal{S}_u$ where $\Delta \not\succeq_u A$ and A contains $f(x)$
		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	let $\langle v, \mathsf{core}', \succ'\rangle = \mathsf{strat}(f, K_1, \mathcal{D})$; if $v \in \mathcal{V}$, then $\succ_v = \succ_v \cap \succ'$,
		otherwise $\mathcal{V} = \mathcal{V} \cup \{v\}$, and $\mathsf{core}_v = \mathsf{core}'$, $\succ_v = \succ'$, and $\mathcal{S}_v = \emptyset$;
		add the edge $\langle u, v, f\rangle$ to $\mathcal{E}$ and $A' \to A'$ to $\mathcal{S}_v$ for each $A' \in K_2 \backslash \mathsf{core}_v$
	where	$\sigma = \{x \mapsto f(x), y \mapsto x\}$, $K_1 = \{A' \in \mathsf{Su}(\mathcal{O}) \mid \top \to A'\sigma \in \mathcal{S}_u\}$
		and $K_2 = \{A' \in \mathsf{Su}(\mathcal{O}) \mid \Gamma' \to \Delta' \vee A'\sigma \in \mathcal{S}_u$ and $\Delta' \not\succeq_u A'\sigma\}$.
Split	If	$\Gamma \to \Delta \vee O(x) \in \mathcal{S}_v$ where $\Delta \not\succeq_v O(x)$
		and $\Gamma' \to \Delta' \vee B(x) \in \mathcal{S}_v$ where $\Delta' \not\succeq_v B(x)$
	then	add $C = \Gamma \wedge \Gamma' \to \Delta \vee \Delta' \vee B(o)$ to $\mathcal{S}_v$, unless $C \,\hat{\in}\, \mathcal{S}_v$ already;
		if $\Gamma = \Gamma' = \top$, and $\Delta = \Delta' = \bot$, add $C' = \top \to B(x)$ to $\mathcal{S}_o$
		unless $C' \,\hat{\in}\, \mathcal{S}_o$ already, where $\mathsf{core}_o = O(x)$;
		else add $C' = B(x) \to B(x)$ to $\mathcal{S}_o$, unless $C' \,\hat{\in}\, \mathcal{S}_o$ already.
Exp	If	$\Gamma \to \Delta \vee O(x) \in \mathcal{S}_v$ where $\Delta \not\succeq_v O(x)$
		and $\bigwedge_{i=1}^{n} A_i \wedge \Gamma' \to \Delta' \in \mathcal{S}_o$, where $\mathsf{core}_o = O(x)$,
		and there is $\Gamma_i \to \Delta_i \vee A_i \in \mathcal{S}_v$, for all $1 \le i \le n$ where $\Delta_i \not\succeq_v A_i$,
	then	add $C = \Gamma \wedge \bigwedge_{i=1}^{n} \Gamma_i \wedge \Gamma'\sigma_o \to \Delta \vee \bigvee_{i=1}^{n} \Delta_i \vee \Delta'$ to $\mathcal{S}_v$,
		unless $C \,\hat{\in}\, \mathcal{S}_v$ already, with $\sigma_o = \{x \mapsto o\}$, where $\pi(o) = O$.
Fct	If	$\Gamma \to \Delta \vee C(o) \vee C(x) \in \mathcal{S}_v$ with $\mathsf{core}_o = O(x)$ s. t. $\pi(o) = O$
	then	add clause $\Gamma \to \Delta \vee C(x)$ to $\mathcal{S}_v$
Join	If	$\Gamma \to \Delta \vee B(o) \in \mathcal{S}_q$ with $\Delta \not\succeq_q B(o)$
		and $B(o) \wedge \Gamma' \to \Delta' \in \mathcal{S}_q$, where $q = r(\mathcal{V})$.
	then	add $C = \Gamma \wedge \Gamma' \to \Delta \vee \Delta'$ to $\mathcal{S}_q$, unless $C \,\hat{\in}\, \mathcal{S}_q$ already.
Root	If	$\Gamma \to \Delta \in \mathcal{S}_v$ is ground or
		$\mathsf{core}_o = O(x)$ for $O \in \Sigma^O$, and Δ contains only unary atoms,
	then	add $C = (\Gamma \to \Delta)\sigma_o$ to $\mathcal{S}_q$, where $q = r(\mathcal{V})$, unless $C \,\hat{\in}\, \mathcal{S}_v$ already;
	where	$\sigma_o = \{x \mapsto o\}$ with $o = \pi(O)$.

Theorem 2 (Completeness). *Let $\mathcal{O}$ be a normalised $\mathcal{SHOI}$ ontology, and $\mathcal{D}$ a context structure which is sound for $\mathcal{O}$ and such that no rule of Table 3 can be applied to it. Given a query clause $\Gamma_Q \to \Delta_Q$ with $\mathsf{core}_q = \Gamma_Q$, we have that $\Gamma_Q \to \Delta_Q \,\hat{\in}\, \mathcal{S}_q$ holds if:*

C1 $\mathcal{O} \models \Gamma_Q \to \Delta_Q$.
C2 For each context atom $A(x) \in \Delta_Q$ and each context atom $A'(x)$ such that $A(x) \succ_q A'(x)$, we have $A'(x) \in \Delta_Q$.
C3 For each $A \in \Gamma_Q$, we have $\Gamma_Q \to A \,\hat{\in}\, \mathcal{S}_q$.

Proposition 1 (Worst-case optimality). *For any expansion strategy introducing a number of contexts exponentially bounded with the size of the ontology, the consequence-based calculus presented in this section works in exponential time.*

Indeed, the number of possible atoms that the algorithm can generate is quadratic on the signature size (in binary atoms, only one term has function symbols), so the number of possible clauses in a context is exponential on the size of $\mathcal{O}$. Since there is an exponential maximum number of contexts, at most exponentially many clauses can be generated. Since each inference uses a quadratic number of clauses, there are at most exponentially many inferences, which shows that the algorithm works in exponential time. Given that subsumption is ExpTime-complete for $\mathcal{SHOI}$ [17], the algorithm is worst-case optimal.

Proposition 2 (Pay-as-you-go behaviour). *For $\mathcal{ELHO}$ ontologies and queries of the form $B_1(x) \to B_2(x)$, the calculus runs in polynomial time with the eager strategy.*

Indeed, with the eager strategy, all derived clauses are of the form $\top \to B(x)$, and since there are no universal quantifiers, a number of contexts at most quadratic is derived. The nominal rules simply exchange clauses between contexts if $\top \to O(x)$ is derived, similarly to the $\mathcal{ELHO}$ algorithm in [10].

6 Conclusion

We have presented the first worst-case optimal consequence-based calculus for $\mathcal{SHOI}$, a description logic with disjunction, inverse roles, and nominals, using explicit ground inferences to exchange information non-locally between contexts. The calculus becomes analogous to known procedures for $\mathcal{ALC}$ and $\mathcal{ELHO}$ whenever the ontology is restricted to those DLs. Further work will consider the extension of the technique presented here to counting quantifiers, which will pose a greater challenge since $\mathcal{SHOIQ}$ is known to be NExpTime-complete.

References

1. Baader, F., Brandt, S., Lutz, C.: Pushing the $\mathcal{EL}$ envelope. In: Proc. of the 19th Int. Joint Conf. on Artificial Intelligence (IJCAI 2005). pp. 364–369 (2005)

2. Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P.F. (eds.): The Description Logic Handbook: Theory, Implementation and Applications. Cambridge University Press (2003)
3. Baader, F., Nipkow, T.: Tern Rewriting and All That. Cambridge University Press (1998)
4. Bate, A., Motik, B., Cuenca Grau, B., Simancik, F., Horrocks, I.: Extending consequence-based reasoning to SRIQ. In: Proc. of the 15th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR 2016). pp. 187–196. AAAI Press (2016), `download/2016/BateMGSH16.pdf`
5. Cuenca Grau, B., Horrocks, I., Motik, B., Parsia, B., Patel-Schneider, P., Sattler, U.: OWL 2: The next step for OWL. J. of Web Semantics 6(4), 309–322 (November 2008), `download/2008/CHMP+08.pdf`
6. Demri, S., de Nivelle, H.: Deciding Regular Grammar Logics with Converse Through First-Order Logic. Journal of Logic, Language and Information (2005)
7. Haarslev, V., Möller, R.: RACER system description. In: Proc. of the Int. Joint Conf. on Automated Reasoning (IJCAR 2001). Lecture Notes in Artificial Intelligence, vol. 2083, pp. 701–705. Springer (2001)
8. Hitzler, P., Vrandečić, D.: Resolution-based approximate reasoning for OWL DL. In: Proc. of the 4th International Semantic Web Conference (ISWC 2005). pp. 383–397 (2005)
9. Kazakov, Y.: Consequence-driven reasoning for Horn SHIQ ontologies. In: Proc. of the 21st Int. Joint Conf. on Artificial Intelligence (IJCAI 2009). pp. 2040–2045 (2009)
10. Kazakov, Y., Krötzsch, M., Simanvcík, F.: Practical Reasoning with Nominals in the EL Family of Description Logics. Proceedings of the Thirteenth International Conference on Principles of Knowledge Representation and Reasoning (2012)
11. Kazakov, Y., Motik, B.: A resolution-based decision procedure for $\mathcal{SHOIQ}$. In: Proc. of the Int. Joint Conf. on Automated Reasoning (IJCAR 2006). Lecture Notes in Artificial Intelligence, vol. 4130, pp. 662–677. Springer (2006)
12. Kharlamov, E., Jiménez-Ruiz, E., Pinkel, C., Rezk, M., Skjæveland, M.G., Soylu, A., Xiao, G., Zheleznyakov, D., Giese, M., Horrocks, I., Waaler, A.: Optique: Ontology-based data access platform. In: International Semantic Web Conference (Posters & Demos). CEUR (`http://ceur-ws.org/`), vol. 1486 (2015), `http://ceur-ws.org/Vol-1486/paper_24.pdf`
13. Linh Anh Nguyen: ExpTime tableaux with global state caching for the description logic SHIO. Neurocomputing (146), 249–263 (2014)
14. Motik, B.: Reasoning in Description Logics using Resolution and Deductive Databases. Ph.D. thesis, Univesität Karlsruhe (TH), Karlsruhe, Germany (January 2006)
15. Motik, B., Shearer, R., Horrocks, I.: Hypertableau reasoning for description logics. J. of Artificial Intelligence Research 36, 165–228 (2009), `download/2009/MoSH09a.pdf`
16. Ortiz, M., Rudolph, S., Simkus, M.: Worst-Case Optimal Reasoning for the Horn-DL Fragments of OWL 1 and 2. Proceedings of the Twelfth International Conference on the Principles of Knowledge Representation and Reasoning (2010)
17. Sattler, U., Vardi, M.Y.: The hybrid μ-calculus. In: Proc. of the Int. Joint Conf. on Automated Reasoning (IJCAR 2001). pp. 76–91 (2001)
18. Schmidt, R.A., Hustadt, U.: A principle for incorporating axioms into the first-order translation of modal formulae. In: Proc. of the 19th Int. Conf. on Automated Deduction (CADE 2002). Lecture Notes in Artificial Intelligence, vol.

2741, pp. 412–426. Springer (2003), \url{http://www.cs.man.ac.uk/~schmidt/publications/SchmidtHustadt03b.html}
19. Simančík, F., Kazakov, Y., Horrocks, I.: Consequence-based reasoning beyond Horn ontologies. In: Proc. of the 22nd Int. Joint Conf. on Artificial Intelligence (IJCAI 2011). pp. 1093–1098 (2011), download/2011/SiKH11a.pdf
20. Simančík, F., Motik, B., Horrocks, I.: Consequence-based and fixed-parameter tractable reasoning in description logics. Artificial Intelligence 209, 29–77 (2014), download/2014/SiMH14a.pdf
21. Sirin, E., Parsia, B., Cuenca Grau, B., Kalyanpur, A., Katz, Y.: Pellet: A practical OWL-DL reasoner. J. of Web Semantics 5(2), 51–53 (2007), http://www.mindswap.org/papers/PelletJWS.pdf
22. Spackman, K.: Managing clinical terminology hierarchies using algorithmic calculation of subsumption: Experience with SNOMED-RT. J. of the Amer. Med. Informatics Ass. (2000), fall Symposium Special Issue
23. Steigmiller, A., Liebig, T., Glimm, B.: Konclude: System description. J. of Web Semantics 27(1) (2014), http://www.websemanticsjournal.org/index.php/ps/article/view/366
24. Tena Cucala, D., Cuenca Grau, B., Horrocks, I.: Consequence-Based Reasoning for Description Logics with Disjunction and Nominals. Tech. rep., University of Oxford, http://bit.ly/2pY2UrN (2017)
25. Tsarkov, D., Horrocks, I.: FaCT++ description logic reasoner: System description. In: Proc. of the Int. Joint Conf. on Automated Reasoning (IJCAR 2006). Lecture Notes in Artificial Intelligence, vol. 4130, pp. 292–297. Springer (2006), download/2006/TsHo06a.pdf
26. Wos, L., Robinson, G.: Paramodulation and theorem-proving in first-order theories with equality. In: Michie, D., Meltzer, B. (eds.) Proceedings of the 4th Annual Machine Intelligence Workshop. Edinburgh University Press (1969)