# Consequence-Based Reasoning for $\mathcal{SHIQ}$

František Simančík and Andrew Bate

Department of Computer Science, University of Oxford, UK

**Abstract.** We propose a novel consequence-based algorithm for TBox reasoning in $\mathcal{SHIQ}$. This work is at very early stage: we have no experimental results and we have not even proved completeness. Instead, we focus on explaining the intuitions behind the algorithm, and show that it has all the favorable properties of existing consequence-based algorithms, namely optimal worst-case complexity, one-pass classification, and pay-as-you-go behavior.

## 1 Background

While many general description logic (DL) [15] reasoners, such as FaCT++ [25], HermiT [18], Pellet [23], and RacerPro [10], are based on variants of highly-optimized tableau algorithms [26], consequence-based (CB) algorithms have received increased attention in recent years due to their favorable properties for TBox reasoning, especially for ontology classification, including optimal worst-case complexity, one-pass classification, and pay-as-you-go behavior [13,21]. CB algorithms were first proposed for (extensions of) $\mathcal{EL}$ [1], then for Horn-$\mathcal{SHIQ}$ [13] and Horn-$\mathcal{SROIQ}$ [19], and recently even for non-Horn $\mathcal{ALCH}$ [21] and $\mathcal{ALCI}$ [22]. Popular CB reasoners for $\mathcal{EL}$ include CEL [3], ELK [14], jcel [17], and Snorocket [16], and prototype implementations for more expressive logics exist and perform well in practice [13,21]. We here propose a novel CB algorithm for TBox reasoning in $\mathcal{SHIQ}$.

## 2 Motivation

CB algorithms can be described in terms of the following principles. Firstly, the algorithms do not build models, instead, they apply inference rules to derive logical *consequences* of the ontology. Secondly, the derived consequences are not stored altogether in one bag, instead, they are distributed amongst multiple *contexts* each corresponding to some logical type in the underlying logic. Finally, the set of contexts is dynamic; new contexts are introduced whenever new types are needed to satisfy existential and universal restrictions occurring in existing contexts. Before we present our CB algorithm for $\mathcal{SHIQ}$ in the next section, here we illustrate these principles on the simpler DL $\mathcal{ALCI}$. The algorithm below is inspired by earlier works (e.g., [21,22]), but we adapt the presentation by making contexts first-class citizens of inference rules.

The CB algorithm for $\mathcal{ALCI}$ derives *clauses* of the form $K \sqsubseteq M$ where $K$ is a conjunction of atomic concepts and $M$ a disjunction of *literals* of the form $A$, $\exists R.A$, or $\forall R.A$ for an atomic concept $A$ and a (possibly inverse) role $R$. In this case, the logical types are conjunctions of atomic concepts, and each context $v$ is associated with a conjunction

$\mathsf{core}_v$ of atomic concepts called the *core* of $v$. The set of clauses stored in context $v$ is denoted $\mathcal{S}_v$; the algorithm maintains that $K = \mathsf{core}_v$ for each clause $K \sqsubseteq M \in \mathcal{S}_v$.[1]

The algorithm takes on input a normalized ontology $\mathcal{O}$ that only contains clauses, and, for classification, it starts with one context $v_A$ with $\mathsf{core}_{v_A} = A$ for each atomic concept $A$ occurring in $\mathcal{O}$. Then, the algorithm applies four inference rules which we here call Core, Hyper, Succ, and Pred. The Core rule simply initializes each set $\mathcal{S}_v$ with the tautological clauses $\mathsf{core}_v \sqsubseteq A$ for each $A \in \mathsf{core}_v$. The Hyper rule applies hyperresolution [20,4] within a single context as follows:

$$\mathsf{Hyper}: \frac{\{\mathsf{core}_v \sqsubseteq M_i \sqcup A_i \in \mathcal{S}_v\}_{i=1}^n \qquad \bigsqcap_{i=1}^n A_i \sqsubseteq M \in \mathcal{O}}{\mathsf{core}_v \sqsubseteq \bigsqcup_{i=1}^n M_i \sqcup M \in \mathcal{S}_v}.$$

Thus, the rule takes $n$ premises from the same context $v$, resolves them with a clause from the ontology, and puts the conclusion back into $v$. The Succ rule introduces a new "successor" context $u$, if needed, for each type induced by one existential and any number of universal restrictions in a "predecessor" context $v$ as follows:

$$\mathsf{Succ}: \frac{\mathsf{core}_v \sqsubseteq M_0 \sqcup \exists R.A \in \mathcal{S}_v \qquad \{\mathsf{core}_v \sqsubseteq M_i \sqcup \forall R.B_i \in \mathcal{S}_v\}_{i=1}^n}{\text{ensure there is a context } u \text{ with } \mathsf{core}_u = A \sqcap \bigsqcap_{i=1}^n B_i}.$$

Observe that the rule can reuse any existing context $u$ with an identical core. For any such pair $\langle v, u\rangle$ of contexts, the Pred rule then resolves the clauses in $v$ with each relevant conclusion in $u$ as follows (note the similarity to the Hyper rule):

$$\mathsf{Pred}: \frac{\begin{array}{l}\mathsf{core}_v \sqsubseteq M_0 \sqcup \exists R.A \in \mathcal{S}_v \\ \{\mathsf{core}_v \sqsubseteq M_i \sqcup \forall R.B_i \in \mathcal{S}_v\}_{i=1}^n\end{array} \qquad A \sqcap \bigsqcap_{i=1}^n B_i \sqsubseteq \bigsqcup_{j=1}^m \forall R^-.C_j \in \mathcal{S}_u}{\mathsf{core}_v \sqsubseteq \bigsqcup_{i=0}^n M_i \sqcup \bigsqcup_{j=1}^m C_j \in \mathcal{S}_v}.$$

The algorithm is complete in the sense that, for each context $v$ and each disjunction of $\underline{\text{atomic}}$ concepts $M$, if $\mathcal{O} \models \mathsf{core}_v \sqsubseteq M$, then the algorithm will eventually derive $\mathsf{core}_v \sqsubseteq M' \in \mathcal{S}_v$ for at least one $M' \subseteq M$. Note that the latter clause entails the former; we say that the latter clause is a *strengthening* of the former. Furthermore, the algorithm terminates in exponential time: it needs to introduce at most exponentially many contexts (one per conjunction), and in each context it can derive at most exponentially many clauses. This is worst-case optimal since reasoning in $\mathcal{ALCI}$ is ExpTime-complete [24].

It is interesting to compare the CB algorithm to tableau- and resolution-based approaches. Similarly to tableau algorithms, the CB algorithm introduces new contexts in a top-down manner as new logical types are discovered. However, unlike tableau algorithms, the CB algorithm can freely reuse an existing context with an identical core, and therefore terminates naturally without need of blocking [11,7] or caching [6,9,8]. Similarly to resolution-based approaches [5,12], the CB algorithm applies inference rules to derive new consequences in a bottom-up manner. Unlike resolution, the CB algorithm avoids consequences about irrelevant types. For example, in $\mathcal{EL}$, where due to lack of universal restrictions the core of each context is just a single atomic concept,

[1] For this reason, some presentations of CB algorithms identify contexts with antecedents of clauses. We will relax this invariant in our algorithm and allow overloading of contexts.

the algorithm only derives clauses $K \sqsubseteq M$ with $|K| = |M| = 1$. In contrast, unrestricted hyperresolution derives clauses with arbitrarily large antecedents even in $\mathcal{EL}$. Furthermore, in the CB algorithm, it is relatively easy to apply the inference rules in different contexts in parallel; this idea underlies the concurrent implementation of ELK [14].

## 3 Algorithm

In this section we present our extension of the CB algorithm to $\mathcal{SHIQ}$. This is non-trivial since, due to the interplay between inverse roles and number restrictions, one may need to consider equality between a successor and the predecessor of an element (in a tree-shaped model). Similarly to pairwise blocking [11] in tableaux, we propose to overcome this difficulty by extending the type of an element $x$ to include its predecessor $y$ and also all roles between $x$ and $y$. Since this is cumbersome in the DL syntax, we use the syntax of first-order logic instead. Thus, instead of a conjunction of atomic concepts, a type is a conjunction of unary and binary atoms over two variables $x$ and $y$, and the algorithm derives first-order clauses with equality over $x$ and $y$. It is helpful to think of $x$ as the *central variable* and of $y$ as the *predecessor variable*. Moreover, we Skolemize existential restrictions and denote the successors of $x$ by *successor terms* $f_i(x)$. We do not, however, use any other function terms and, in particular, no nesting of functions.

**Definition 1 (Variables, Terms, Literals).** *A* variable *is either the* central variable $x$, *the* predecessor variable $y$, *or a* neighbor variable $y_i$ *for* $i \in \mathbb{N}$. *A* successor term *is of the form* $f_i(x)$ *for* $i \in \mathbb{N}$, *with* $f_i$ *a unary function symbol. A* term *is either a variable or a successor term. A* neighbor term *is any term except the central variable* $x$.

*Let* $s$ *and* $t$ *be neighbor terms. An* atom *is an expression of the form* $A(x)$, $A(t)$, $R(x,t)$, *or* $R(t,x)$ *for* $A$ *an atomic concept and* $R$ *an atomic role. A* literal *is either an atom, an* equality $s \approx t$, *or an* inequality $s \not\approx t$. *Equality and inequality are defined to be symmetric operators. We reserve the letters* $T$ *and* $L$ *for atoms and literals, respectively. A literal* $L$ *is* function-free *if no function symbols occur in* $L$.

Notice that, for example, we do not allow literals of the form $R(s,t)$ or $x \approx t$ for neighbor terms $s$ and $t$—these are not needed for $\mathcal{SHIQ}$—and we consider $s \approx t$ and $t \approx s$ to be the same literals. Next we introduce variable substitutions and clauses.

**Definition 2 (Substitutions).** *A substitution* $\sigma$ *is a partial function from variables to terms. Given a substitution* $\sigma$ *and an expression* $E$, *we write* $E\sigma$ *for the result of simultaneously replacing each variable* $v \in \mathrm{dom}(\sigma)$ *in* $E$ *by the term* $\sigma(v)$. *We write* $[t_1/v_1, \dots, t_n/v_n]$ *to denote the substitution* $\sigma$ *for which* $\sigma(v_i) = t_i$ *for each* $1 \leq i \leq n$.

*A substitution* $\sigma$ *is* central *if it maps the central variable* $x$ *to itself, i.e., if* $\sigma(x) = x$.

**Definition 3 (Clauses).** *A* clause *is an implication of the form* $\bigwedge_{i=1}^{n} T_i \rightarrow \bigvee_{j=1}^{m} L_j$, *where* $n, m \geq 0$, *each* $T_i$ *is a function-free atom, and each* $L_j$ *a literal. Clauses have standard first-order semantics with each variable assumed to be universally quantified.*

*We identify conjunctions and disjunctions of literals with sets of literals (i.e., they are unordered and without repeated literals) and we use them in standard set operations; furthermore, we write the empty conjunction and the empty disjunction as* $\top$ *and*

**Table 1.** Example translations from DL axioms to normal clauses

| | | |
|---|---|---|
| $\bigsqcup_{i=1}^{n} A_i \sqsubseteq \bigsqcup_{j=1}^{m} B_j$ | $\bigwedge_{i=1}^{n} A_i(x) \rightarrow \bigvee_{j=1}^{m} B_j(x)$ | (1) |
| $A \sqsubseteq \exists R.B$ | $A(x) \rightarrow R(x, f_k(x))$ and $A(x) \rightarrow B(f_k(x))$ | (2) |
| $A \sqsubseteq \forall R.B$ | $A(x) \wedge R(x, y_1) \rightarrow B(y_1)$ | (3) |
| $A \sqsubseteq {\geqslant} n\, R.B$ | $A(x) \rightarrow R(x, f_i(x))$ and $A(x) \rightarrow B(f_i(x))$ for $1 \leq i \leq n$<br>$A(x) \rightarrow f_i(x) \not\approx f_j(x)$ for $1 \leq i < j \leq n$ | (4) |
| $A \sqsubseteq {\leqslant} n\, R.B$ | $A(x) \wedge \bigwedge_{i=0}^{n}(R(x, y_i) \wedge B(y_i)) \rightarrow \bigvee_{0 \leq i < j \leq n} y_i \approx y_j$ | (5) |
| $R \sqsubseteq S$ | $R(x, y_1) \rightarrow S(x, y_1)$ | (6) |
| $\mathsf{Disjoint}(R, S)$ | $R(x, y_1) \wedge S(x, y_1) \rightarrow \bot$ | (7) |

$\bot$, *respectively. We reserve the letter* $K$ *for a conjunction of function-free atoms, and* $M$ *for a disjunction of literals, thus we often write clauses as* $K \rightarrow M$.

*We say that a clause* $K_1 \rightarrow M_1$ *is a* strengthening *of a clause* $K_2 \rightarrow M_2$ *if* $K_1 \subseteq K_2$ *and* $M_1 \subseteq M_2$*; furthermore, we write* $K \rightarrow M \hat{\in} \mathcal{S}$ *to denote that the set of clauses* $\mathcal{S}$ *contains at least one strengthening of* $K \rightarrow M$.

Our algorithm will only derive clauses over the two variables $x$ and $y$. The neighbor variables $y_i$ can occur only in the input ontology which needs to be normalized as follows. For simplicity, we do not allow the predecessor variable $y$ to occur in ontologies.

**Definition 4 (Normal Clauses).** *A clause* $\alpha$ *is* normal *if* $y$ *does not occur in* $\alpha$*, and each neighbor variable* $y_i$ *that occurs in* $\alpha$ *also occurs in some atom of the form* $R(x, y_i)$ *or* $R(y_i, x)$ *in the antecedent of* $\alpha$*. A* normal ontology *is a finite set of normal clauses.*

Our normal clauses are similar to HT-clauses considered by Motik et al. [18]. By applying the normalization method from that paper and subsequently Skolemizing all at-least restrictions, one can transform an arbitrary $\mathcal{SHIQ}$ TBox $\mathcal{T}$ into a normal ontology $\mathcal{O}$ that entails the same consequences over the atomic concepts occurring in $\mathcal{T}$. Moreover, assuming unary encoding of number restrictions, this can be performed in polynomial time. Example translations are shown in Table 1. Note that normal clauses are more general than what is strictly needed for $\mathcal{SHIQ}$. For example, they can express role disjointness, as in (7), and even arbitrary safe role expressions.

The following definition describes the main datastructure used in our algorithm. Essentially, the algorithm operates with a collection of contexts labeled by types, and it derives two kinds of consequences: clauses in contexts and edges between contexts.

**Definition 5 (Context Structure).** *A* type *is a finite conjunction of function-free atoms over* $x$ *and* $y$*. A* context structure *is a tuple* $\mathcal{D} = \langle \mathcal{V}, \mathsf{core}, \mathcal{S}, \mathcal{E} \rangle$*, where* $\mathcal{V}$ *is a finite set of* contexts*,* $\mathsf{core}$ *is a function that assigns to each context* $v \in \mathcal{V}$ *a type* $\mathsf{core}_v$*,* $\mathcal{S}$ *is a function that assigns to each context* $v \in \mathcal{V}$ *a finite set* $\mathcal{S}_v$ *of clauses over* $x$ *and* $y$*, and* $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V} \times \{f_i\}_{i \in \mathbb{N}}$ *is a finite set of edges labeled by function symbols.*

*Such context structure* $\mathcal{D}$ *is* sound *for a normal ontology* $\mathcal{O}$ *provided:*

- $\mathcal{O} \models \mathsf{core}_v \wedge K \rightarrow M$ *for each context* $v \in \mathcal{V}$ *and each clause* $K \rightarrow M \in \mathcal{S}_v$*, and*
- $\mathcal{O} \models \mathsf{core}_v \rightarrow \mathsf{core}_u[f_k(x)/x, x/y]$ *for each edge* $\langle v, u, f_k \rangle \in \mathcal{E}$.

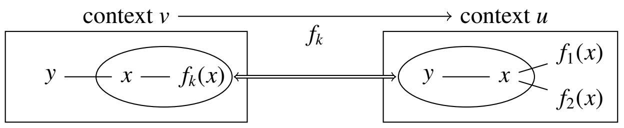

**Fig. 1.** Relationship between contexts $v$ and $u$ connected by an edge $\langle v, u, f_k \rangle$

Observe how the above definition gives semantics to clauses and edges in a context structure. Unlike in Section 2, here we assume that each clause in $\mathcal{S}_v$ implicitly contains $\mathsf{core}_v$ in the antecedent. For example, the clause $\top \rightarrow A(x) \in S_v$ represents the axiom $\mathsf{core}_v \rightarrow A(x)$. An edge $\langle v, u, f_k \rangle$ asserts that each element of type $\mathsf{core}_v$ has an $f_k$-successor of type $\mathsf{core}_u$. As illustrated in Fig. 1, it helps to mentally identify the central variable $x$ and the successor term $f_k(x)$ in context $v$ with the predecessor variable $y$ and the central variable $x$ in context $u$, respectively. For example, if $\mathsf{core}_u = R(y, x) \wedge B(x)$, then the edge $\langle v, u, f_k \rangle$ represents the axiom $\mathsf{core}_v \rightarrow R(x, f_k(x)) \wedge B(f_k(x))$.

To avoid proliferation of contexts, we will restrict types to atoms that can "trigger" hyperresolution with some clause in the input ontology $\mathcal{O}$ as in the next definition.

**Definition 6 (Triggers).** *Let $\mathcal{O}$ be a normal ontology. An atom $T$* occurs negatively in *$\mathcal{O}$ if $T$ occurs in the antecedent of some normal clause in $\mathcal{O}$. An atom $T$ over $x$ and $y$ is a* trigger in *$\mathcal{O}$ if at least one of the following conditions holds:*

- *$T[y_i/y]$ occurs negatively in $\mathcal{O}$ for some neighbor variable $y_i$, or*
- *$T = A(x)$ for some atomic concept $A$ that occurs in $\mathcal{O}$.*

We denote the set of all triggers in $\mathcal{O}$ by triggers($\mathcal{O}$). Intuitively, if $A(y_i)$ occurs negatively in $\mathcal{O}$ for no neighbor variable $y_i$, then it is irrelevant whether a neighbor of an element is an instance of $A$, so it is not necessary to consider the atom $A(y)$ in a type. Indeed, our algorithm will only consider types that are subsets of triggers($\mathcal{O}$). As we will see in Section 4, this leads to nice pay-as-you-go behavior on fragments of $\mathcal{SHIQ}$.

We are now ready to present our CB algorithm. It takes on input a normal ontology $\mathcal{O}$ and a context structure $\mathcal{D}$ sound for $\mathcal{O}$, and it exhaustively applies the inference rules from Table 2, which modify $\mathcal{D}$. Rules Core, Hyper, and Pred are analogous to the corresponding rules in Section 2. The Hyper rule requires a central substitution; in particular, it cannot map $x$ to a successor term $f_i(x)$, so there is no nesting of functions. The Pred rule applies over an edge $\langle v, u, f_k \rangle$ and it only pushes those atoms from $u$ to $v$ that are triggers. The Eq rule applies to an equality $s \approx t$ and an arbitrary literal $L(s)$, and produces the literal $L(t)$ by replacing each occurrence of $s$ in $L(s)$ by $t$. The Ineq rule eliminates inequalities $t \not\approx t$, which are trivially false. The redundancy elimination rule Elim is optional. It can be used to delete a clause from a context if a (strict) strengthening of the clause is derived in the same context. Note that a variant of Elim is already incorporated in the negative preconditions of all inference rules: no rule is applicable if a strengthening of its conclusion is already present in the context.

The Succ rule is more involved than the rule in Section 2 and needs explanation. For a context $v$ and a successor term $f_k(x)$, the rule gathers into type $K_1$ all triggers that are known to hold for the $f_k$-successor, and into type $K_2$ all triggers that might hold for

**Table 2.** Inference rules

| | | |
|---|---|---|
| Core | If | $T \in \mathsf{core}_v$, |
| | | and $\top \to T \notin \mathcal{S}_v$, |
| | then | add $\top \to T$ to $\mathcal{S}_v$. |
| Hyper | If | $\bigwedge_{i=1}^{n} T_i \to M \in \mathcal{O}$, |
| | | $\sigma$ is a central substitution, |
| | | $K_i \to M_i \vee T_i\sigma \in \mathcal{S}_v$ for $1 \leq i \leq n$, |
| | | and $\bigwedge_{i=1}^{n} K_i \to \bigvee_{i=1}^{n} M_i \vee M\sigma \mathrel{\hat{\notin}} \mathcal{S}_v$, |
| | then | add $\bigwedge_{i=1}^{n} K_i \to \bigvee_{i=1}^{n} M_i \vee M\sigma$ to $\mathcal{S}_v$. |
| Succ | If | $f_k(x)$ occurs in $\mathcal{S}_v$ and no edge $\langle v, u, f_k \rangle \in \mathcal{E}$ exists such that |
| | | $T \to T \mathrel{\hat{\in}} \mathcal{S}_u$ for each atom $T \in K_2 \setminus \mathsf{core}_u$, |
| | then | let $\langle u, \mathsf{core}' \rangle := \mathsf{strategy}(K_1, \mathcal{D})$; |
| | | if $u \notin \mathcal{V}$, then let $\mathcal{V} := \mathcal{V} \cup \{u\}$, $\mathsf{core}_u := \mathsf{core}'$, and $\mathcal{S}_u := \emptyset$; |
| | | add the edge $\langle v, u, f_k \rangle$ to $\mathcal{E}$; and |
| | | add $T \to T$ to $\mathcal{S}_u$ for each atom $T \in K_2 \setminus \mathsf{core}_u$; |
| | where | $K_1 = \{T \in \text{triggers}(\mathcal{O}) \mid \top \to T[f_k(x)/x, x/y] \in \mathcal{S}_v\}$ and |
| | | $K_2 = \{T \in \text{triggers}(\mathcal{O}) \mid K' \to M' \vee T[f_k(x)/x, x/y] \in \mathcal{S}_v\}$. |
| Pred | If | $\langle v, u, f_k \rangle \in \mathcal{E}$, |
| | | $\bigwedge_{i=1}^{n} T'_i \to \bigvee_{j=1}^{m} T_j \in \mathcal{S}_u$, |
| | | $K_i \to M_i \vee T'_i[f_k(x)/x, x/y] \in \mathcal{S}_v$ for $1 \leq i \leq n$, |
| | | $T_j[y/x, x/y] \in \text{triggers}(\mathcal{O})$ for $1 \leq j \leq m$, |
| | | and $\bigwedge_{i=1}^{n} K_i \to \bigvee_{i=1}^{n} M_i \vee \bigvee_{j=1}^{m} T_j[f_k(x)/x, x/y] \mathrel{\hat{\notin}} \mathcal{S}_v$, |
| | then | add $\bigwedge_{i=1}^{n} K_i \to \bigvee_{i=1}^{n} M_i \vee \bigvee_{j=1}^{m} T_j[f_k(x)/x, x/y]$ to $\mathcal{S}_v$. |
| Eq | If | $K_1 \to M_1 \vee s \approx t$, |
| | | $K_2 \to M_2 \vee L(s) \in \mathcal{S}_v$, |
| | | and $K_1 \wedge K_2 \to M_1 \vee M_2 \vee L(t) \mathrel{\hat{\notin}} \mathcal{S}_v$, |
| | then | add $K_1 \wedge K_2 \to M_1 \vee M_2 \vee L(t)$ to $\mathcal{S}_v$. |
| Ineq | If | $K \to M \vee t \not\approx t \in \mathcal{S}_v$, |
| | | and $K \to M \mathrel{\hat{\notin}} \mathcal{S}_v$, |
| | then | add $K \to M$ to $\mathcal{S}_v$. |
| Elim | If | $K_1 \to M_1 \in \mathcal{S}_v$, |
| | | $K_2 \to M_2 \in \mathcal{S}_v$, |
| | | $K_1 \to M_1$ is a strengthening of $K_2 \to M_2$ and the two clauses are distinct, |
| | then | remove $K_2 \to M_2$ from $\mathcal{S}_v$. |

the $f_k$-successor. The rule then connects $v$ by an $f_k$-labeled edge to some (either new or existing) context $u$ with $\mathsf{core}_u \subseteq K_1$, and for each atom $T \in K_2 \setminus \mathsf{core}_u$ the rule pushes into $S_u$ the tautology $T \to T$; we say that the rule *overloads* context $u$ with atom $T$. Roughly speaking, in a context $u$ overloaded with atoms $T_1, \ldots, T_n$, the algorithm will derive consequences of all types between $\mathsf{core}_u$ and $\mathsf{core}_u \wedge T_1 \wedge \ldots \wedge T_n$.[2] However, if a context $u$ is overloaded with atoms $T_1$ and $T_2$ in two different applications of the $\mathsf{Succ}$ rule, then the algorithm may derive in $\mathcal{S}_u$ clauses with $T_1 \wedge T_2$ in the antecedent, even though the type $\mathsf{core}_u \wedge T_1 \wedge T_2$ would otherwise not be considered.

Thus, context overloading allows more aggressive reuse of contexts at the cost of reasoning about irrelevant types. Intuitively, there is a trade-off between (i) eagerly introducing many specific contexts with large cores, and (ii) heavily overloading few generic contexts with small cores. In the extreme case, the $\mathsf{Succ}$ rule can always reuse just one context $u$ with the empty core and overload it with all atoms. Since different levels of context overloading may lead to very different algorithmic behavior, we do not hardcode any fixed strategy for choosing the context $u$ and its core in our $\mathsf{Succ}$ rule. Instead, we formulate the rule in Table 2 with a generic expansion strategy as follows.

**Definition 7 (Strategy).** *An* expansion strategy *is a polynomial-time computable function* $\mathsf{strategy}$ *that takes on input a type $K$ and a context structure $\mathcal{D} = \langle \mathcal{V}, \mathsf{core}, \mathcal{S}, \mathcal{E} \rangle$. The result of* $\mathsf{strategy}(K, \mathcal{D})$ *is a pair $\langle v, \mathsf{core}' \rangle$ where $\mathsf{core}'$ is a subset of $K$, and either $v \notin \mathcal{V}$ is a new context or $v \in \mathcal{V}$ is an existing context in $\mathcal{D}$ such that $\mathsf{core}_v = \mathsf{core}'$.*

We propose three concrete strategies. We reserve for each type $K$ a distinguished context $v_K$ whose core will be set to $K$. On input $K$ and $\mathcal{D}$, the *eager* strategy returns $\langle v_K, K \rangle$, the *central* strategy returns $\langle v_{K^x}, K^x \rangle$ where $K^x = \{A(x) \in K\}$, and the *trivial* strategy returns $\langle v_\top, \top \rangle$. These strategies can return at most exponentially many different contexts, in which case the algorithm terminates in exponential time. However, strategies that return multiple contexts with the same core are also possible.

The algorithm is sound as in Theorem 1. We propose that the algorithm is complete for deriving strengthenings of clauses over $x$ as in Conjecture 2; this is similar to what we had for the algorithm in Section 2, but we have not produced a full proof yet.

**Theorem 1 (Soundness).** *Let $\mathcal{O}$ be a normal ontology and let $\mathcal{D}$ be a context structure that is sound for $\mathcal{O}$. Inference rules from Table 2 preserve soundness of $\mathcal{D}$ for $\mathcal{O}$.*

*Proof.* Let $\mathcal{O}$ be a normal ontology and let $\mathcal{D} = \langle \mathcal{V}, \mathsf{core}, \mathcal{S}, \mathcal{E} \rangle$ be an arbitrary context structure that is sound for $\mathcal{O}$. We show that each context structure obtained by applying an inference rule from Table 2 to $\mathcal{D}$ is also sound for $\mathcal{O}$. In the proof we rely on soundness of hyperresolution: for arbitrary formulas $\omega$, $\phi_i$, $\psi_i$, and $\gamma_i$, where $1 \leq i \leq n$, we have

$$\bigwedge_{j=1}^{n} \phi_j \to \omega \text{ and } \gamma_i \to \psi_i \vee \phi_i \text{ for } 1 \leq i \leq n \text{ imply } \bigwedge_{i=1}^{n} \gamma_i \to \bigvee_{i=1}^{n} \psi_i \vee \omega. \tag{8}$$

($\mathsf{Core}$ rule) If $T \in \mathsf{core}_v$, then clearly $\mathcal{O} \models \mathsf{core}_v \to T$.

[2] This idea is related to "context partitioning" from [21].

(Hyper rule) Suppose (i) $\bigwedge_{i=1}^{n} T_i \rightarrow M \in \mathcal{O}$ and (ii) $K_i \rightarrow M_i \vee T_i\sigma \in \mathcal{S}_v$ for $1 \leq i \leq n$, where $\sigma$ is a central substitution. It follows from (i) that $\mathcal{O} \models \bigwedge_{i=1}^{n} T_i\sigma \rightarrow M\sigma$. Since $\mathcal{D}$ is sound for $\mathcal{O}$, (ii) implies $\mathcal{O} \models \mathsf{core}_v \wedge K_i \rightarrow M_i \vee T_i\sigma$ for $i \leq i \leq n$. By (8), we conclude $\mathcal{O} \models \mathsf{core}_v \wedge \bigwedge_{i=1}^{n} K_i \rightarrow \bigvee_{i=1}^{n} M_i \vee M\sigma$, as required.

(Succ rule) Let $\sigma := [f_k(x)/x, x/y]$. Since $\mathcal{D}$ is sound for $\mathcal{O}$, we have $\mathcal{O} \models \mathsf{core}_v \rightarrow T\sigma$ for each atom $T \in K_1$; hence $\mathcal{O} \models \mathsf{core}_v \rightarrow K_1\sigma$. Since $\mathsf{core}_u \subseteq K_1$ by the requirement on strategy in Definition 7, also $\mathcal{O} \models \mathsf{core}_v \rightarrow \mathsf{core}_u\sigma$. Therefore, it is sound to add the edge $\langle v, u, f_k\rangle$ to $\mathcal{E}$. Finally, it is clearly sound to add the tautological clauses $T \rightarrow T$ to $\mathcal{S}_u$ in the last line of the rule.

(Pred rule) Let $\sigma := [f_k(x)/x, x/y]$. Suppose (i) $\bigwedge_{i=1}^{n} T_i' \rightarrow \bigvee_{j=1}^{m} T_j \in \mathcal{S}_u$, (ii) $\langle v, u, f_k\rangle \in \mathcal{E}$, and (iii) $K_i \rightarrow M_i \vee T_i'\sigma \in \mathcal{S}_v$ for $1 \leq i \leq n$. Since $\mathcal{D}$ is sound for $\mathcal{O}$, (i) implies $\mathcal{O} \models \mathsf{core}_u\sigma \wedge \bigwedge_{i=1}^{n} T_i'\sigma \rightarrow \bigvee_{j=1}^{m} T_j\sigma$, (ii) implies $\mathcal{O} \models \mathsf{core}_v \rightarrow \mathsf{core}_u\sigma$, and (iii) implies $\mathcal{O} \models \mathsf{core}_v \wedge K_i \rightarrow M_i \vee T_i'\sigma$ for $1 \leq i \leq n$. By (8), we conclude $\mathcal{O} \models \mathsf{core}_v \wedge \bigwedge_{i=1}^{n} K_i \rightarrow \bigvee_{i=1}^{n} M_i \vee \bigvee_{j=1}^{m} T_j\sigma$, as required.

The Eq and the Ineq rules are trivially sound. Finally, for the Elim rule, it is always sound to remove a clause from a context. □

**Conjecture 2 (Completeness)** *Let $\mathcal{O}$ be a normal ontology and let $\mathcal{D} = \langle \mathcal{V}, \mathsf{core}, \mathcal{S}, \mathcal{E}\rangle$ be a context structure such that no inference rule from Table 2 is applicable to $\mathcal{O}$ and $\mathcal{D}$. Let $v \in \mathcal{V}$ be a context and let $\alpha$ be a clause of the form $\bigwedge_i A_i(x) \rightarrow \bigvee_j B_j(x)$ such that $\mathsf{core}_v \subseteq \bigwedge_i A_i(x)$ and $A_i(x) \rightarrow A_i(x) \hat{\in} \mathcal{S}_v$ for each i. If $\mathcal{O} \models \alpha$, then $\alpha \hat{\in} \mathcal{S}_v$.*

For classification, we initialize the algorithm with the context $v_{A(x)}$ with core $A(x)$ for each atomic concept $A$; the algorithm will then derive in $v_{A(x)}$ all entailed clauses of the form $A(x) \rightarrow \bigvee_j B_j(x)$. Alternatively, one could initialize the algorithm with only the context $v_\top$ with core $\top$ and overload it with atoms $A(x)$ for all atomic concept $A$; the algorithm will then derive in $v_\top$ all entailed clauses of the form $\bigwedge_i A_i(x) \rightarrow \bigvee_j B_j(x)$.

*Example 1.* Let $\mathcal{O}$ be the ontology consisting of the following normal clauses:

$$A(x) \rightarrow R(x, f_1(x)) \tag{9}$$

$$R(x, y_1) \rightarrow B(x) \vee C(y_1) \tag{10}$$

$$C(x) \rightarrow D(x) \tag{11}$$

$$D(x) \wedge R(y_1, x) \rightarrow S(x, y_1) \vee B(y_1) \tag{12}$$

$$R(x, y_1) \wedge S(y_1, x) \rightarrow \bot \tag{13}$$

Our goal is to prove that $\mathcal{O} \models A(x) \rightarrow B(x)$. First, we identify the triggers in $\mathcal{O}$:

$$\mathsf{triggers}(\mathcal{O}) = \{A(x), R(x, y), B(x), C(x), D(x), R(y, x), S(y, x)\}.$$

Note that $S(x, y)$ is not a trigger because no $S(x, y_i)$ occurs in the antecedent of a clause. We initialize the algorithm with a context structure $\mathcal{D} = \langle \mathcal{V}, \mathsf{core}, \mathcal{S}, \mathcal{E}\rangle$ with $\mathcal{V} = \{v\}$, $\mathcal{E} = \emptyset$, $\mathsf{core}_v = A(x)$, and $\mathcal{S}_v = \emptyset$. We then derive the following clauses in $\mathcal{S}_v$:

$$\top \rightarrow A(x) \quad \text{by Core from } \mathsf{core}_v \tag{14}$$

$$\top \rightarrow R(x, f_1(x)) \quad \text{by Hyper applied to (9) and (14)} \tag{15}$$

$$\top \rightarrow B(x) \vee C(f_1(x)) \quad \text{by Hyper applied to (10) and (15)} \tag{16}$$

Note that we cannot apply Hyper to (11) and (16) to derive $\top \rightarrow B(x) \vee D(f_1(x))$ because the necessary substitution is not central, i.e., we cannot map $x$ to $f_1(x)$. Instead, we now apply Succ to $f_1(x)$ in $v$ using the eager strategy. Since we have $K_1 = \{R(y, x)\}$ and $K_2 = \{C(x), R(y, x)\}$ in the Succ rule, the rule adds a new context $u$ to $\mathcal{V}$ with $\mathsf{core}_u = R(y, x)$, adds the edge $\langle v, u, f_1\rangle$ to $\mathcal{E}$, and overloads $u$ with $C(x)$. We find that $\mathcal{S}_u$ contains the clauses:

| | | |
|---|---|---|
| $C(x) \rightarrow C(x)$ | overloading in the Succ rule above | (17) |
| $\top \rightarrow R(y, x)$ | by Core from $\mathsf{core}_u$ | (18) |
| $C(x) \rightarrow D(x)$ | by Hyper applied to (11) and (17) | (19) |
| $C(x) \rightarrow S(x, y) \vee B(y)$ | by Hyper applied to (12) and (18) and (19) | (20) |

Observe now that both $S(y, x)$ and $B(x)$ are triggers. Hence, we can apply Pred over the edge $\langle v, u, f_1\rangle \in \mathcal{E}$ and derive the following clauses in $\mathcal{S}_v$:

| | | |
|---|---|---|
| $\top \rightarrow B(x) \vee S(f_1(x), x)$ | by Pred applied to (20) and (16) | (21) |
| $\top \rightarrow B(x)$ | by Hyper applied to (13) and (15) and (21) | (22) |

Hence $\mathcal{O} \models A(x) \rightarrow B(x)$ since the algorithm is sound and we have $\mathsf{core}_v = A(x)$ and $\top \rightarrow B(x) \in \mathcal{S}_v$. If we had chosen to use the trivial or the central strategy instead of the eager strategy in the above, then $\mathsf{core}_u$ would have been empty and $u$ would have been overloaded with both $C(x)$ and $R(y, x)$. The clauses derived by the algorithm would have been the same, except that (18) and (20) would also contain $R(y, x)$ in the antecedent.

## 4 Pay-As-You-Go Behavior

In this section we discuss the behavior of our algorithm on various fragments of normal clauses. We start with the Horn fragment (a clause is Horn if it contains at most one literal in the consequent). All inference rules in Table 2 are Horn-preserving, so on a Horn ontology the algorithm only derives Horn clauses. Furthermore, with the eager strategy on a Horn ontology, there is no overloading of contexts and the algorithm only derives clauses with empty antecedents; therefore, there are at most polynomially many clauses in each context. This might not be the case for other strategies where, due to context overloading, the algorithm can derive Horn clauses with larger antecedents.

Next, we consider various DL fragments of $\mathcal{SHIQ}$. Since the Succ only constructs types consisting of triggers in $\mathcal{O}$, we have the following properties:

1. If, for an atomic concept $A$, there is no negative occurrence of $A(y_i)$ in $\mathcal{O}$ for a neighbor variable $y_i$, then the algorithm never puts $A(y)$ in the core of a context.
2. If, for an atomic role $R$, there is no occurrence of $R(t, x)$ in $\mathcal{O}$ for a neighbor term $t$, then the algorithm never puts $R(x, y)$ nor $R(y, x)$ in the core of a context.

Observe (e.g., in Table 1) that negative atoms $A(y_i)$ are needed only for qualified at-most restrictions, and atoms $R(t, x)$ are needed only for inverse roles. In particular, none of these are needed for DLs such as $\mathcal{SHN}$, in which case all cores are simply conjunctions of atoms $A(x)$. This is similar to switching from pairwise to single blocking in tableau algorithms [18], but in our algorithm it happens automatically by restricting types to triggers. Note, however, that the behavior of the algorithm depends not only on the underlying DL, but also on the particular translation from DL axioms to normal clauses. This is illustrated in the example below.

*Example 2.* Consider a DL ontology $\mathcal{O} = \{A \sqsubseteq \exists R.\top, A \sqsubseteq \forall R.B\}$. The first axiom is Skolemized to (23) below, but the second axiom can be written either as (24) or (25).

$$A(x) \to R(x, f_1(x)) \tag{23}$$

$$A(x) \wedge R(x, y_1) \to B(y_1) \tag{24}$$

$$A(y_1) \wedge R(y_1, x) \to B(x) \tag{25}$$

Let $\mathcal{O}_1 = \{(23), (24)\}$ and $\mathcal{O}_2 = \{(23), (25)\}$. Then $\text{triggers}(\mathcal{O}_1) = \{A(x), B(x), R(x, y)\}$ and $\text{triggers}(\mathcal{O}_2) = \{A(x), B(x), A(y), R(y, x)\}$. Thus, although (24) and (25) are logically equivalent, the negative occurrence of $A(y_1)$ in (25) suggests that it is important to know if the predecessor of an element is an instance of $A$, whereas the negative occurrence of $A(x)$ in (24) suggests that it is important to know whether the element itself is an instance of $A$. To illustrate this, consider a context $v$ with $\mathsf{core}_v = A(x)$. In either case, the $\mathsf{Core}$ and the $\mathsf{Hyper}$ rules successively derive the following clauses in context $v$:

$$\top \to A(x) \in \mathcal{S}_v \quad \text{and} \quad \top \to R(x, f_1(x)) \in S_v. \tag{26}$$

First, consider the case of $\mathcal{O}_1$. The $\mathsf{Hyper}$ rule resolves the clauses in (26) with (24) to derive $\top \to B(f_1(x)) \in \mathcal{S}_v$. The $\mathsf{Context}$ rule with the eager strategy then introduces a new context $u_1$ whose core contains each atom $T \in \text{triggers}(\mathcal{O}_1)$ such that $\top \to T[f_1(x)/x, x/y] \in \mathcal{S}_v$. Thus $\mathsf{core}_{u_1} = B(x)$, and $\top \to B(x) \in \mathcal{S}_{u_1}$ by the $\mathsf{Core}$ rule.

Second, consider the case of $\mathcal{O}_2$. The $\mathsf{Hyper}$ rule cannot resolve (26) with (25) since a central substitution cannot map $x$ to $f_1(x)$. Instead, the $\mathsf{Context}$ rule with the eager strategy now introduces a new context $u_2$ with $\mathsf{core}_{u_2} = A(y) \wedge R(y, x)$. The conclusion $\top \to B(x) \in \mathcal{S}_{u_2}$ is then derived by applying the $\mathsf{Hyper}$ rule in $u_2$.

Thus, in both cases we arrived at a new context whose core implies $B(x)$, but the two cores are different. Moreover, in case of $\mathcal{O}_1$ the axiom $A \sqsubseteq \forall R.B$ fired in context $v$ between the central variable $x$ and the successor term $f_1(x)$, whereas in case of $\mathcal{O}_2$ the axiom fired in the new context $u_2$ between the predecessor and the central variables.

Finally, we consider the behavior of our algorithm on $\mathcal{EL}$ ontologies. These are often normalized to only contain axioms of the form $\bigsqcap_i A_i \sqsubseteq B$, $A \sqsubseteq \exists R.B$, and $\exists R.A \sqsubseteq B$ [1]. The first two forms are translated to normal clauses straightforwardly as in Table 1, but the last form deserves discussion. One can first rewrite the axiom to its negation normal form $\top \sqsubseteq B \sqcup \forall R.\neg A$, and this then to the normal clauses $R(x, y_1) \to B(x) \vee \bar{A}(y_1)$ and $A(x) \wedge \bar{A}(x) \to \bot$. These are, unfortunately, not Horn. Instead, it is more common in CB reasoning to treat the axiom $\exists R.A \sqsubseteq B$ as the equivalent $A \sqsubseteq \forall R^{-}.B$ [13] which, although it introduces an inverse role, rewrites to the Horn clause $A(x) \wedge R(y_1, x) \to B(y_1)$. We will assume this second translation in the rest of the discussion.

In that case, the set of triggers contains apart from all central atoms $A(x)$ also the atom $R(y, x)$ for each role $R$ occurring in some negative existential restriction. Therefore, apart from one context per atomic concept, with the eager strategy the algorithm will also introduce one context with core $R(y,x) \wedge B(x)$ for each positive existential restriction $\exists R.B$ such that $R$ also occurs in some negative existential restriction. Although these additional contexts are not usually considered by standard $\mathcal{EL}$ algorithms, their number is still only linear in the size of the ontology. Therefore, with the eager strategy, our algorithm terminates in polynomial time on $\mathcal{EL}$ ontologies.

Notably, including the incoming role in the type of a context has been considered before for supporting role range axioms in $\mathcal{EL}$ [2]. Indeed, we can write a range axiom $\top \sqsubseteq \forall R.A$ as $R(y_1,x) \to A(x)$; the algorithm will then derive $\top \to A(x)$ in each context whose core contains the incoming role $R(y,x)$. We leave it as an interesting exercise for the reader to figure out what would happen if we wrote the axiom as $R(x,y_1) \to A(y_1)$.

If desired, standard $\mathcal{EL}$ algorithms can be recovered by switching to the central strategy. In that case, there will be only one context per atomic concept. Although the Succ rule applied to an existential restriction $\exists R.C$ can (if $R$ also occurs in some negative existential restriction) overload the context $v$ whose core is $C(x)$ with atom $R(y,x)$, the additional clause $R(y,x) \to R(y,x)$ can only participate in the Hyper rule as follows:

$$\frac{\top \to A(x) \in \mathcal{S}_v \qquad R(y,x) \to R(y,x) \in \mathcal{S}_v \qquad A(x) \wedge R(y_1,x) \to B(y_1) \in \mathcal{O}}{R(y,x) \to B(y) \in \mathcal{S}_v}.$$

The conclusion of the rule can then participate in the Pred rule to push $B(x)$ to predecessors of context $v$. This is analogous to deriving axiom $\mathsf{core}_v \sqsubseteq \forall R^-.B$ in the algorithm in Section 2. Since there can be only polynomially many such axioms, our algorithm terminates in polynomial time on $\mathcal{EL}$ ontologies even with the central strategy. With the trivial strategy (reusing just one context), on the other hand, the algorithm may derive clauses with arbitrarily large antecedents, and hence potentially run in exponential time.

## 5 Future Work

Being pay-as-you-go w.r.t. $\mathcal{EL}$, our algorithm has potential to perform competitively on ontologies that are largely $\mathcal{EL}$ but contain a few more expressive constructors. This has been confirmed experimentally for other CB reasoners [13,21]. Implementation and evaluation of the algorithm is future work. It may well turn out that further optimizations will be needed to make the algorithm really practical. One optimization commonly considered in resolution is to impose a partial order over literals and restrict rule application to maximal literals [4,21]; it should be investigated how this affects completeness.

Our normal clauses are sufficient for $\mathcal{SHIQ}$ and also, e.g., for safe Boolean expressions on roles. For technical reasons, we have omitted reflexive roles—atoms $R(x,x)$—from the definition since, due to number restrictions, these can lead to equality even between the central and the predecessor variable. We do not, however, expect any fundamental difficulties there. In contrast, since the complexity of reasoning in $\mathcal{SHOIQ}$ rises to NExpTime [24], extending the algorithm to nominals might be difficult.

## References

1. Baader, F., Brandt, S., Lutz, C.: Pushing the $\mathcal{EL}$ envelope. In: Kaelbling, L., Saffiotti, A. (eds.) Proc. 19th Int. Joint Conf. on Artificial Intelligence (IJCAI'05). pp. 364–369. Professional Book Center (2005)
2. Baader, F., Brandt, S., Lutz, C.: Pushing the $\mathcal{EL}$ envelope further. In: Clark, K.G., Patel-Schneider, P.F. (eds.) Proc. OWLED 2008 DC Workshop on OWL: Experiences and Directions. CEUR Workshop Proceedings, vol. 496. CEUR-WS.org (2008)
3. Baader, F., Lutz, C., Suntisrivaraporn, B.: CEL—a polynomial-time reasoner for life science ontologies. In: Furbach, U., Shankar, N. (eds.) Proc. 3rd Int. Joint Conf. on Automated Reasoning (IJCAR'06). LNCS, vol. 4130, pp. 287–291. Springer (2006)
4. Bachmair, L., Ganzinger, H.: Resolution theorem proving. In: Robinson, J.A., Voronkov, A. (eds.) Handbook of Automated Reasoning, pp. 19–99. Elsevier and MIT Press (2001)
5. de Nivelle, H., Schmidt, R.A., Hustadt, U.: Resolution-based methods for modal logics. Logic J. of the IGPL 8(3), 265–292 (2000)
6. Donini, F.M., Massacci, F.: EXPTIME tableaux for $\mathcal{ALC}$. Artificial Intelligence 124(1), 87–138 (2000)
7. Glimm, B., Horrocks, I., Motik, B.: Optimized description logic reasoning via core blocking. In: Giesl, J., Hähnle, R. (eds.) Proc. 5th Int. Joint Conf. on Automated Reasoning (IJCAR'10). LNAI, vol. 6173, pp. 457–471. Springer (2010)
8. Goré, R., Nguyen, L.A.: EXPTIME tableaux with global caching for description logics with transitive roles, inverse roles and role hierarchies. In: Olivetti, N. (ed.) Proc. 16th Int. Conf. on Automated Reasoning with Analytic Tableaux and Related Methods. LNCS, vol. 4548, pp. 133–148. Springer (2007)
9. Goré, R., Widmann, F.: Sound global state caching for $\mathcal{ALC}$ with inverse roles. In: Giese, M., Waaler, A. (eds.) Proc. 18th Int. Conf. on Automated Reasoning with Analytic Tableaux and Related Methods. LNCS, vol. 5607, pp. 205–219. Springer (2009)
10. Haarslev, V., Möller, R.: Racer system description. In: Goré, R., Leitsch, A., Nipkow, T. (eds.) Proc. 1st Int. Joint Conf. on Automated Reasoning (IJCAR'01). LNCS, vol. 2083, pp. 701–705. Springer (2001)
11. Horrocks, I., Sattler, U.: A description logic with transitive and inverse roles and role hierarchies. J. of Logic and Computation 9(3), 385–410 (1999)
12. Hustadt, U., Motik, B., Sattler, U.: Deciding expressive description logics in the framework of resolution. Information and Computation 206(5), 579–601 (2008)
13. Kazakov, Y.: Consequence-driven reasoning for Horn $\mathcal{SHIQ}$ ontologies. In: Boutilier, C. (ed.) Proc. 21st Int. Joint Conf. on Artificial Intelligence (IJCAI'09). pp. 2040–2045. IJCAI (2009)
14. Kazakov, Y., Krötzsch, M., Simančík, F.: Concurrent classification of $\mathcal{EL}$ ontologies. In: Aroyo, L., Welty, C., Alani, H., Taylor, J., Bernstein, A., Kagal, L., Noy, N., Blomqvist, E. (eds.) Proc. 10th Int. Semantic Web Conf. (ISWC'11). LNCS, vol. 7032, pp. 305–320. Springer (2011)
15. Krötzsch, M., Simančík, F., Horrocks, I.: A description logic primer. CoRR abs/1201.4089 (2012)
16. Lawley, M.J., Bousquet, C.: Fast classification in Protégé: Snorocket as an OWL 2 EL reasoner. In: Taylor, K., Meyer, T., Orgun, M. (eds.) Proc. 6th Australasian Ontology Workshop (IAOA'10). Conferences in Research and Practice in Information Technology, vol. 122, pp. 45–49. Australian Computer Society Inc. (2010)
17. Mendez, J.: jcel: A modular rule-based reasoner. In: Horrocks, I., Yatskevich, M., Jimenez-Ruiz, E. (eds.) Proc. OWL Reasoner Evaluation Workshop 2012 (ORE'12). CEUR Workshop Proceedings, vol. 858. CEUR-WS.org (2012)

18. Motik, B., Shearer, R., Horrocks, I.: Hypertableau reasoning for description logics. J. of Artificial Intelligence Research 36, 165–228 (2009)
19. Ortiz, M., Rudolph, S., Simkus, M.: Worst-case optimal reasoning for the Horn-DL fragments of OWL 1 and 2. In: Lin, F., Sattler, U., Truszczynski, M. (eds.) Proc. 12th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR'10). pp. 269–279. AAAI Press (2010)
20. Robinson, J.A.: Automatic deduction with hyper-resolution. Int. J. of Computer Mathematics 1, 227–234 (1965)
21. Simančík, F., Kazakov, Y., Horrocks, I.: Consequence-based reasoning beyond Horn ontologies. In: Walsh, T. (ed.) Proc. 22nd Int. Joint Conf. on Artificial Intelligence (IJCAI'11). pp. 1093–1098. AAAI Press/IJCAI (2011)
22. Simančík, F., Motik, B., Krötzsch, M.: Fixed parameter tractable reasoning in DLs via decomposition. In: Rosati, R., Rudolph, S., Zakharyaschev, M. (eds.) Proc. 24th Int. Workshop on Description Logics (DL'11). CEUR Workshop Proceedings, vol. 745, pp. 400–410. CEUR-WS.org (2011)
23. Sirin, E., Parsia, B., Grau, B.C., Kalyanpur, A., Katz, Y.: Pellet: A practical OWL-DL reasoner. J. of Web Semantics 5(2), 51–53 (2007)
24. Tobies, S.: Complexity Results and Practical Algorithms for Logics in Knowledge Representation. Ph.D. thesis, RWTH Aachen, Germany (2001)
25. Tsarkov, D., Horrocks, I.: FaCT++ description logic reasoner: System description. In: Furbach, U., Shankar, N. (eds.) Proc. 3rd Int. Joint Conf. on Automated Reasoning (IJCAR'06). LNCS, vol. 4130, pp. 292–297. Springer (2006)
26. Tsarkov, D., Horrocks, I., Patel-Schneider, P.F.: Optimizing terminological reasoning for expressive description logics. J. of Automated Reasoning 39(3), 277–316 (2007)