# Optimized DL Reasoning via Core Blocking

Birte Glimm, Ian Horrocks, and Boris Motik

Oxford University Computing Laboratory, UK

## 1 Introduction

State of the art reasoners for expressive DLs are typically model building procedures that decide the (un)satisfiability of a knowledge base $\mathcal{K}$ via a constructive search for an abstraction of a model for $\mathcal{K}$. Despite numerous optimizations, certain existing and emerging knowledge bases still pose significant challenges to such reasoners mainly because these abstractions can be very large.

To ensure that only finite model abstractions are constructed (hyper)tableau reasoners use a cycle detection technique called *blocking*. It has already been demonstrated that using a more fine-grained blocking condition can make the constructed abstractions smaller, resulting in a significant speedup [1]. Even with such a blocking condition, however, the constructed model abstractions can be very large; furthermore, checking such fine-grained conditions can itself be costly.

To address these problems, we propose a new *core blocking* technique. Our technique first employs an easy-to-check and very "aggressive" blocking condition that can halt the model construction much earlier than existing techniques. This condition is so aggressive that, if used alone, it is not necessarily the case that the constructed abstraction can be expanded into a model. Therefore, after a model abstraction has been constructed, a detailed check is performed to ensure that all blocks are indeed valid, and the model construction terminates only if all blocks pass this check.

We further present an empirical evaluation using a prototypical implementation of our technique in the HermiT reasoner. The evaluation compares the performance of the hypertableau algorithm employing the original blocking condition and several core blocking variants on widely used ontologies. The evaluation shows that the model abstraction size can be reduced significantly. The effects of core blocking are most pronounced with large and complex ontologies such as DOLCE or GALEN. Furthermore, core blocking allows HermiT to classify an OWL version of the FMA ontology [2], whereas with standard blocking the reasoner runs out or memory.

Further details and evaluation results are available in a technical report [3].

## 2 Preliminaries

The formal definition of the hypertableau calculus is technically involved; therefore, we will introduce only those aspects needed to understand the idea behind core blocking. For further details and the precise definitions, we refer to [4].

The calculus is applicable to a knowledge base $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ expressed in $\mathcal{SROIQ}$ [5]. The calculus does not operate on $\mathcal{K}$ directly; rather, in order to reduce nondeterminism, it first translates $\mathcal{K}$ into a set of clauses $\mathcal{C}$ and an ABox $\mathcal{A}$. The class of clauses on which the hypertableau calculus operates is called HT-clauses. An HT-clause is an implication of the form $\bigwedge_{i=1}^{m} U_i \rightarrow \bigvee_{j=1}^{n} V_j$, where $U_i$ and $V_j$ are called the *antecedent* and the *consequent* atoms, respectively. Most notably, for $r$ an atomic role, $s$ is a role, $A$ is an atomic concept, and $B$, each *antecedent* atom is of the form $A(x)$, $r(x,x)$, $r(x,y_i)$, $r(y_i,x)$, $A(y_i)$, or $A(z_j)$. Each *consequent* atom is of the form $B(x)$, $\geqslant n\, s.B(x)$, $B(y_i)$, $r(x,x)$, $r(x,y_i)$, $r(y_i,x)$, $r(x,z_j)$, $r(z_j,x)$, $x \approx z_j$, or $y_i \approx y_j$. These syntactic restrictions reflect the structure of DL axioms and ultimately ensure termination of the calculus. HT-clauses are straightforwardly interpreted in first-order logic, and they intuitively state that at least one consequent atom must be true whenever all atoms in the antecedent are true. We next revise the derivation rules of the calculus.

The Hyp-rule is the main derivation rule. The rule is applicable to an HT-clause $cl$ and an ABox $\mathcal{A}_\ell$ if a mapping $\sigma$ from the variables in $cl$ to the individuals in $\mathcal{A}_\ell$ exists such that $\sigma(U_i) \in \mathcal{A}_\ell$ for each $1 \leq i \leq m$, but $\sigma(V_j) \notin \mathcal{A}_\ell$ for each $1 \leq j \leq m$; if such $\sigma$ exists, then a consequent atom $V_j$ of $cl$ is nondeterministically chosen and $\mathcal{A}_\ell$ is extended to $\mathcal{A}_{\ell+1} = \mathcal{A}_\ell \cup \{\sigma(V_j)\}$. For example, when applied to the HT-clause $r(x,y) \rightarrow (\geqslant 1\, r.A)(x) \vee D(y)$ and an ABox $\mathcal{A}_\ell$ containing $r(a,b)$, the Hyp-rule extends $\mathcal{A}_\ell$ either with $(\geqslant 1\, r.A)(a)$ or $D(b)$. The $\geqslant$-rule deals with existential quantifiers and number restrictions. Let $\mathsf{ar}(s,a,b) = s(a,b)$ if $s$ is an atomic role and $\mathsf{ar}(s,a,b) = r(b,a)$ if $s$ is an inverse role such that $s = r^-$. The rule is applicable to $(\geqslant n\, r.B)(a) \in \mathcal{A}_\ell$ if no individuals $b_1, \ldots, b_n$ exist such that $\mathsf{ar}(r,a,b_i) \in \mathcal{A}_\ell$ and $B(b_i) \in \mathcal{A}_\ell$ for each $1 \leq i \leq n$, and $b_i \not\approx b_j \in \mathcal{A}_\ell$ for each $1 \leq i < j \leq n$. If this is the case, then $\mathcal{A}_\ell$ is extended to $\mathcal{A}_{\ell+1}$ by introducing fresh individuals $c_1, \ldots, c_n$ and adding assertions $\mathsf{ar}(r,a,c_i)$ and $B(c_i)$ for $1 \leq i \leq n$, and $c_i \not\approx c_j$ for $1 \leq i < j \leq n$.

The $\approx$-rule deals with equality: given $a \approx b$, the rule replaces the individual $a$ in all assertions with the individual $b$, and adds some bookkeeping information to keep track of the rule application. Finally, the $\perp$-rule detects contradictions—called *clashes*—such as $A(a)$ and $\neg A(a)$, or $a \not\approx a$. A clash-free ABox to which no derivation rule is applicable is called a *pre-model*.

## 2.1 Blocking

Unrestricted application of the $\geqslant$-rule could lead to nontermination of the HT calculus. To prevent that, the $\geqslant$-rule is applied to an assertion $(\geqslant n\, r.B)(a)$ only if the individual $a$ is not *blocked*, as described next.

To apply blocking, the individuals are split into two sets. *Root* individuals (mainly individuals occurring in the input), which are never blocked and *blockable* individuals, which are introduced by the $\geqslant$-rule and they can be blocked. For $A$ an atomic concept and $r$ an atomic role, we define labels of an individual and an individual pair as follows:

$$\mathcal{L}_\mathcal{A}(s) = \{A \mid A(s) \in \mathcal{A}\} \qquad \mathcal{L}_\mathcal{A}(s,t) = \{r \mid r(s,t) \in \mathcal{A}\}$$

To prevent cyclic blocks, we use a strict order $\prec$ over all individuals, which coincides with the order in which individuals are inserted into the ABox.

*Pairwise anywhere blocking* is necessary for knowledge bases that use inverse roles and number restrictions. Each individual $s$ in an ABox $\mathcal{A}$ is assigned by induction on $\prec$ a status as follows: $s$ is *blocked* if it is directly or indirectly blocked; $s$ is *indirectly* *blocked* if it has a blocked ancestor; and $s$ is *directly blocked* by an individual $t$ if, for $s'$ and $t'$ the predecessors of $s$ and $t$, respectively, $s$, $t$, $s'$, and $t'$ are all blockable, $t$ is not blocked, $t \prec s$, and $(1)$ –$(4)$ hold.

$$\mathcal{L}_{\mathcal{A}}(s) = \mathcal{L}_{\mathcal{A}}(t) \qquad (1) \qquad\qquad \mathcal{L}_{\mathcal{A}}(s') = \mathcal{L}_{\mathcal{A}}(t') \qquad (2)$$
$$\mathcal{L}_{\mathcal{A}}(s, s') = \mathcal{L}_{\mathcal{A}}(t, t') \qquad (3) \qquad\qquad \mathcal{L}_{\mathcal{A}}(s', s) = \mathcal{L}_{\mathcal{A}}(t', t) \qquad (4)$$

For an efficient implementation, we build, for each individual, a blocking signature that consist of the four label sets. A hash table containing the blocking signatures for possible blockers can then be used to cheaply look-up a blocker for an unblocked individual before the $\geqslant$-rule is applied.

The simpler *single anywhere blocking* can be used on knowledge bases without inverse roles, and it differs from the above definition in that $s$ is *directly blocked* by an individual $t$ if $s$ and $t$ are blockable, $t$ is not blocked, $t \prec s$, and $(1)$ holds.

A pre-model $\mathcal{A}'$ can be extended to a model for $(\mathcal{A}, \mathcal{C})$ by unraveling. Roughly speaking, each individual $s$ that is directly blocked in $\mathcal{A}'$ by $t$ is replaced by a "copy" of $t$; a precise account of this process is given in [4].

## 3 Optimized Blocking Strategies

For tableau algorithms that normally require pairwise blocking, Horrocks and Sattler proposed a more precise blocking condition [1], which amounts to single subset blocking with additional constraints on the predecessor of the individual that is to be blocked and on the blocker itself. Although checking the blocking conditions is quite expensive, the optimization exhibits substantial improvements in reasoning performance due to the significantly smaller pre-models.

Related blocking optimizations were proposed in the context of first-order theorem proving [6]; however, these techniques do not guarantee termination for DLs such as $\mathcal{SROIQ}$ that provide for nominals, number restrictions, and inverse roles.

Caching [7] is an orthogonal approach for reducing the pre-model size by reusing already constructed pre-model fragments. In fact, caching techniques can be used to obtain a worst-case optimal algorithm for certain DLs [8, 9]; in contrast, standard (hyper)tableau algorithms are usually not worst-case optimal.

### 3.1 Core Blocking

Unlike existing blocking techniques, core blocking is approximate rather than exact: applying core blocking alone does not guarantee that a pre-model can indeed be unraveled into a model. To ensure the latter, a pre-model needs to be checked to discover invalid blocks; if such blocks are found, the derivation is continued until either a contradiction is derived or all blocks become valid.

To formalize the process of discovering approximate blocks, we assume that each assertion $\alpha$ in an ABox is associated with a Boolean flag that determines whether $\alpha$ is a *core assertion*. A *core blocking policy* will be used to determine which assertions are core. In Section 3.3 we present two policies that strike a balance between the potential for reduction in the pre-model size and the cost of validating blocks. Before that, however, we introduce a general notion of core blocking that is applicable to any policy.

**Definition 1.** *For an ABox $\mathcal{A}$ and a pair of individuals $s$ and $t$, let*

$$\begin{aligned}\mathcal{L}^{\mathsf{core}}_{\mathcal{A}}(s) &= \{A \mid A \in \mathcal{L}_{\mathcal{A}}(s) \text{ and } A(s) \text{ is a core assertion in } \mathcal{A}\} \text{ and}\\ \mathcal{L}^{\mathsf{core}}_{\mathcal{A}}(s,t) &= \{r \mid r(s,t) \in \mathcal{L}_{\mathcal{A}}(s,t) \text{ and } r(s,t) \text{ is a core assertion in } \mathcal{A}\}.\end{aligned}$$

Single *and* pairwise core blocking *are obtained from the respective definitions given in Section 2.1 by using $\mathcal{L}^{\mathsf{core}}_{\mathcal{A}}$ instead of $\mathcal{L}_{\mathcal{A}}$ in conditions (1)–(4); furthermore, in single core blocking, for $s$ to be directly blocked by $t$ we additionally require both $s$ and $t$ to be successors of blockable individuals.*

The requirement that $s$ and $t$ are successors of blockable individuals ensures that single core blocking can also be used with knowledge bases that contain inverse roles.

A blocking validation test checks whether any of the derivation rules would be applicable if we were to unravel a candidate pre-model $\mathcal{A}_\ell$ to a model. If no rule becomes applicable, then we can guarantee that the model construction succeeds, i.e., the block is indeed valid. To this end, we define an ABox $\mathsf{val}_{\mathcal{A}_\ell}(s)$ for a blockable individual $s$ that, intuitively, contains the assertions from the unraveling of $\mathcal{A}_\ell$ that affect inferences involving $s$.

**Definition 2.** *Let $\mathcal{C}$ be a set of HT clauses, and let $\mathcal{A}_\ell$ be an ABox. For an individual $w$, let $|w| = w$ if $w$ is not blocked in $\mathcal{A}_\ell$, and $|w| = w'$ if $w$ is blocked in $\mathcal{A}_\ell$ by $w'$. For a blockable individual $s$, the ABox $\mathsf{val}_{\mathcal{A}_\ell}(s)$ is the union of the sets shown in the following table, where $u$ denotes the predecessor of $s$, $v$ denotes a successor of $|s|$, $b$ denotes a root individual, $C$ denotes a concept, and $r$ denotes an atomic role.*

| 1 | 2 | 3 |
|---|---|---|
| $\{C(u) \mid C(u) \in \mathcal{A}_\ell\}$ | $\{r(u,s) \mid r(u,s) \in \mathcal{A}_\ell\}$ | $\{r(s,u) \mid r(s,u) \in \mathcal{A}_\ell\}$ |
| $\{C(s) \mid C(\lvert s\rvert) \in \mathcal{A}_\ell\}$ | | |
| $\{C(v) \mid C(\lvert v\rvert) \in \mathcal{A}_\ell\}$ | $\{r(s,v) \mid r(\lvert s\rvert,v) \in \mathcal{A}_\ell\}$ | $\{r(v,s) \mid r(v,\lvert s\rvert) \in \mathcal{A}_\ell\}$ |
| $\{C(b) \mid C(b) \in \mathcal{A}_\ell\}$ | $\{r(s,b) \mid r(\lvert s\rvert,b) \in \mathcal{A}_\ell\}$ | $\{r(b,s) \mid r(b,\lvert s\rvert) \in \mathcal{A}_\ell\}$ |

*A blockable individual $s$ is* safe for blocking *in an ABox $\mathcal{A}_\ell$ if the following conditions are satisfied:*

- *the* Hyp*-rule is not applicable to an HT-clause $\gamma \in \mathcal{C}$ and $\mathsf{val}_{\mathcal{A}_\ell}(s)$ with a mapping $\sigma$ such that $\sigma(x) = s$, and*
- *the $\geqslant$-rule is not applicable to an assertion $(\geqslant n\, r.B)(s)$ in $\mathsf{val}_{\mathcal{A}_\ell}(s)$.*

*A directly blocked individual $s$ with predecessor $s'$ is* validly blocked *in $\mathcal{A}_\ell$ if both $s$ and $s'$ are safe for blocking.*

On knowledge bases that normally require single blocking (i.e., that do not contain inverse roles), Definitions 1 and 2 can be simplified. By the model construction from [4], $\mathsf{val}_{\mathcal{A}_\ell}(s)$ then needs to contain only sets from columns 1 and 2 in Definition 2; this, in turn, allows us to drop the extra requirement on the predecessors of $s$ and $t$ in Definition 1 in the case of single core blocking.

## 3.2 Applying Core Blocking in a Derivation

If an individual $s$ is core-blocked by an individual $t$ but the block is identified as invalid, one should reconsider $t$ as a potential blocker for $s$ only after $\mathsf{val}_{\mathcal{A}_\ell}(s)$ changes; otherwise, the calculus might get stuck in an endless loop trying to block $s$ by $t$ and subsequently discovering the block to be invalid. We deal with this problem by associating with each individual $s$ in $\mathcal{A}_\ell$ a Boolean flag $\mathsf{mod}_{\mathcal{A}_\ell}(s)$ that is updated as the derivation progresses. Intuitively, $\mathsf{mod}_{\mathcal{A}_\ell}(s) = \mathsf{true}$ means that $\mathsf{val}_{\mathcal{A}_\ell}(s)$ has changed since the last time blocks were checked for validity. We also maintain a set $S$ of pairs of validly blocked and blocking individuals, which we to ensure that the calculus terminates only when all blocks are valid.

**Definition 3.** *Let $S$ be a set of pairs of individuals; let $\mathcal{A}_\ell$ be an ABox; and let $s$ and $t$ be individuals occurring in $\mathcal{A}_\ell$. Then, $s$ is directly blocked by $t$ in $\mathcal{A}_\ell$ for $S$-*core blocking *iff $s$ is directly blocked by $t$ in $\mathcal{A}_\ell$ for core blocking and*

$$\langle s,t\rangle \in S \qquad \textit{or} \qquad \mathsf{mod}_{\mathcal{A}_\ell}(s) = \mathsf{true} \qquad \textit{or} \qquad \mathsf{mod}_{\mathcal{A}_\ell}(t) = \mathsf{true}.$$

*A derivation by the* hypertableau calculus with core blocking *for a set of HT-clauses $\mathcal{C}$ and an ABox $\mathcal{A}$ is constructed by applying the following steps.*

1. *Set $S := \emptyset$, $\mathcal{A}^{\mathsf{a}} := \mathcal{A}$, and $\mathsf{mod}_{\mathcal{A}^{\mathsf{a}}}(s) := \mathsf{true}$ for each individual $s$ in $\mathcal{A}^{\mathsf{a}}$.*
2. *Apply the hypertableau calculus exhaustively to $\mathcal{A}^{\mathsf{a}}$ and $\mathcal{C}$ while using $S$-core blocking in the $\geqslant$-rule; furthermore, whenever $\mathcal{A}_{\ell+1}$ is derived from $\mathcal{A}_\ell$, for each individual $s$ in $\mathcal{A}_{\ell+1}$ set*
   *(a)* $\mathsf{mod}_{\mathcal{A}_{\ell+1}}(s) := \mathsf{true}$ *if* $\mathsf{val}_{\mathcal{A}_{\ell+1}}(s) \neq \mathsf{val}_{\mathcal{A}_\ell}(s)$ *or if $s$ does not occur in $\mathcal{A}_\ell$, and*
   *(b)* $\mathsf{mod}_{\mathcal{A}_{\ell+1}}(s) := \mathsf{mod}_{\mathcal{A}_\ell}(s)$ *otherwise.*
   *Let $\mathcal{A}^{\mathsf{b}}$ be a resulting ABox to which no derivation rule is applicable.*
3. *Set $S$ to be equal to the set of pairs $\langle s,t\rangle$ of individuals such that $s$ is directly blocked in $\mathcal{A}^{\mathsf{b}}$ by $t$ and $s$ is validly blocked in $\mathcal{A}^{\mathsf{b}}$.*
4. *Set $\mathsf{mod}_{\mathcal{A}^{\mathsf{b}}}(s) := \mathsf{false}$ for each individual $s$ in $\mathcal{A}^{\mathsf{b}}$.*
5. *If an individual $s$ exists such that $s$ is core blocked in $\mathcal{A}^{\mathsf{b}}$ by $t$ but $\langle s,t\rangle \notin S$, then set $\mathcal{A}^{\mathsf{a}} := \mathcal{A}^{\mathsf{b}}$ and go to Step 2.*
6. *Return $\mathcal{A}^{\mathsf{b}}$.*

Roughly speaking, our algorithm first applies the derivation rules as usual, with the difference that core blocking is used (this is because $S = \emptyset$ in Step 1). After computing a candidate pre-model $\mathcal{A}^{\mathsf{b}}$ in Step 2, in Step 3 the algorithm updates $S$ to the set of pairs of valid blocks, and in Step 4 it marks all individuals in $\mathcal{A}^{\mathsf{b}}$ as not changed. In Step 5, the algorithm checks whether $\mathcal{A}^{\mathsf{b}}$ contains invalid blocks. If that is the case, the process is repeated; but then, $S$-core blocking ensures that only those blocks are considered that are known to be valid or for which at least one of the individuals has changed since the last validation. Theorem 1 shows that the calculus is sound, complete, and terminating.

**Theorem 1.** *Let $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ be a $\mathcal{SROIQ}$ knowledge base and $\mathcal{C}$ the set of HT-clauses for $\mathcal{K}$.*

1. *The hypertableau calculus with core blocking terminates.*

2. *If $\mathcal{C}$ and $\mathcal{A}$ are satisfiable, then $\bot \notin \mathcal{A}^{\mathsf{b}}$ for some $\mathcal{A}^{\mathsf{b}}$ computed by the calculus.*
3. *If $\mathcal{C}$ and $\mathcal{A}$ are unsatisfiable, then $\bot \in \mathcal{A}^{\mathsf{b}}$ for each $\mathcal{A}^{\mathsf{b}}$ computed by the calculus.*

*Proof (Sketch).* For the first claim, assume that $\mathcal{A}^{\mathsf{b}}$ is an ABox computed in Step 2 such that, whenever $s$ is directly blocked in $\mathcal{A}^{\mathsf{b}}$ by $t$ for core blocking, then $s$ is directly blocked in $\mathcal{A}^{\mathsf{b}}$ by $t$ for standard blocking. Each individual $s$ is then validly blocked in $\mathcal{A}^{\mathsf{b}}$, so $\langle s, t\rangle \in S$ at Step 3 and the condition at Step 5 is not satisfied, so the calculus terminates. Thus, in the worst case, core blocking reduces to standard blocking, which implies a bound on the size of $\mathcal{A}^{\mathsf{b}}$ in the usual way [4]. Furthermore, if an individual $t$ does not validly block $s$ in an ABox $\mathcal{A}^{\mathsf{b}}$, then $t$ can be considered again as a blocker for $s$ only after $\mathsf{val}_{\mathcal{A}^{\mathsf{b}}}(s)$ or $\mathsf{val}_{\mathcal{A}^{\mathsf{b}}}(t)$ changes. Since $\mathcal{A}^{\mathsf{b}}$ is bounded in size, $\mathsf{val}_{\mathcal{A}^{\mathsf{b}}}(s)$ and $\mathsf{val}_{\mathcal{A}^{\mathsf{b}}}(t)$ can change only a bounded number of times; hence, $t$ is considered as a candidate blocker for $s$ only a finite number of times, which implies termination.

The second claim holds in the same way as in [4]. Finally, for the third claim, given an ABox $\mathcal{A}^{\mathsf{b}}$ computed by the calculus such that $\bot \notin \mathcal{A}^{\mathsf{b}}$, we unravel $\mathcal{A}^{\mathsf{b}}$ into an interpretation in the standard way [4]. From the definition of unraveling in [4], one can see that, for each blockable individual $s$, the ABox $\mathsf{val}_{\mathcal{A}^{\mathsf{b}}}(s)$ contains the assertions that correspond to the part of the unraveled interpretation involving $s$. Since $s$ is validly blocked in $\mathcal{A}^{\mathsf{b}}$, all the relevant restrictions are satisfied for $s$. Since all blocks are valid in $\mathcal{A}^{\mathsf{b}}$, the unraveled interpretation is a model of $\mathcal{C}$ and $\mathcal{A}$. □

## 3.3 Core Blocking Policies

We now present two policies for identifying core assertions. Each policy can be used with either single or pairwise core blocking.

The *simple core policy* is inspired by the following observation. Let $\mathcal{A}$ be a potentially infinite ABox obtained by applying the hypertableau calculus without blocking to an $\mathcal{EL}$ knowledge base $\mathcal{K}$, and let $s$ and $t$ be two individuals introduced by applying the $\geqslant$-rule to assertions of the form $(\geqslant n\, r.B)(s')$ and $(\geqslant m\, r'.B)(t')$. Then, $\mathcal{L}_{\mathcal{A}}(s) = \mathcal{L}_{\mathcal{A}}(t)$; in fact, the concept labels $\mathcal{L}_{\mathcal{A}}(s)$ and $\mathcal{L}_{\mathcal{A}}(t)$ depend only on the concept $B$. The policy thus makes such assertions $B(s)$ and $B(t)$ core in the hope that, if a knowledge base is sufficiently "$\mathcal{EL}$-like," then $s$ would validly block $t$.

**Definition 4.** *The* simple core *policy marks all assertions as not core unless they are covered by one of the following rules.*

- *Each assertion $B(c_j)$ derived by applying the $\geqslant$-rule to an assertion of the form $(\geqslant n\, r.B)(a)$ is marked as core.*
- *Each assertion $\alpha'$ derived by the $\approx$-rule from an assertion $\alpha$ via merging is marked as core if and only if $\alpha$ is core.*
- *If an ABox contains $\alpha$ as a noncore assertion but some derivation rule derives $\alpha$ as a core assertion, the former assertion is replaced with the latter.*

Simple core blocking generates very small cores, but it can be imprecise and can therefore lead to frequent validation of blocks. For example, if $s$ and $t$ are individuals introduced by applying the $\geqslant$-rule as above, then inferences involving the predecessor of $s$ can cause the propagation of new concepts to $s$, which might invalidate blocking.

Furthermore, if the knowledge base contains nondeterministic concepts, then nondeterministic inferences involving $s$ and $t$ may cause $\mathcal{L}_{\mathcal{A}}(s)$ and $\mathcal{L}_{\mathcal{A}}(t)$ to diverge, which can also invalidate blocking. We therefore define the following, stronger notion of cores.

**Definition 5.** *The* complex core *policy is the extension of the simple core policy in which, whenever the* Hyp*-rule derives an assertion $\sigma(V_j)$ using a mapping $\sigma$ and an HT-clause $\gamma = \bigwedge_{i=1}^{m} U_i \rightarrow \bigvee_{j=1}^{n} V_j$, the assertion $\sigma(V_j)$ is marked as core if and only if $\sigma(V_j)$ is a concept assertion and*

- $n > 1$, *or*
- $\sigma(V_j)$ *is of the form $B(s)$ with $s$ a successor of $\sigma(w)$ for some variable $w$ in $\gamma$.*

The complex core policy is motivated by the fact that, when $\mathcal{EL}$-style algorithms are extended to expressive but deterministic DLs such as Horn-$\mathcal{SHIQ}$ [10], the concepts that are propagated to an individual from its predecessor uniquely determine the individual's label, so we mark all such assertions as core.

## 4 Empirical Evaluation

We implemented the different core blocking strategies in our HermiT reasoner and carried out a preliminary empirical evaluation. For the evaluation, we selected several ontologies commonly used in practice. We classified each ontology and tested the satisfiability of all concepts from the ontologies with the different blocking strategies. Our main measurement is the number individuals in the final pre-model since this number directly relates to the amount of memory required by the reasoner.

We conducted our tests on a 2.6 GHz Windows 7 Desktop machine with 8 GB of RAM. We used Java 1.6 allowing for 1 GB of heap space in each test. All tested ontologies, a version of HermiT that supports core blocking, and Excel spreadsheets containing test results are available online.[1]

Figures 1–4 contain concepts on the x-axis; however, concept names are not shown due to the high number of concepts. The concepts are ordered according to the performance under the standard blocking strategy reasoner. The y-axis either displays the number of individuals in the pre-models or the reasoning times in milliseconds. All reasoning times exclude loading and preprocessing times, since these are independent of the blocking strategy. Some figures employ a logarithmic scale to improve readability. The label *standard pairwise* refers to the standard pairwise anywhere blocking strategy, *complex pairwise* refers to pairwise core blocking with the complex core policy, etc.

Tables 1–2 show average measurements taken while testing the satisfiability of all concepts in an ontology. The meaning of various rows is as follows: *final pre-model size* shows the average number of individuals in the final pre-model; *finally blocked* shows the average number of blocked individuals in the final pre-model; and *number of validations* shows the average number of validations before a pre-model was found in which all blocks are valid; *time in ms* shows the average time to test concept satisfiability; and *validation part* shows the percentage of this time taken to validate blocks. Finally, all tables show the time needed to classify the ontology in the format *hours:minutes:seconds*.

[1] http://www.hermit-reasoner.com/coreBlocking.html

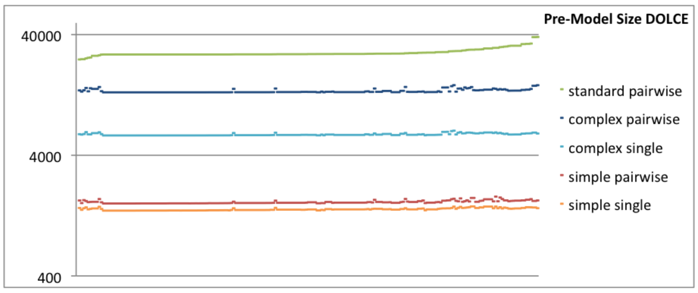

**Fig. 1.** The number of individuals in the pre-models for all concepts in DOLCE

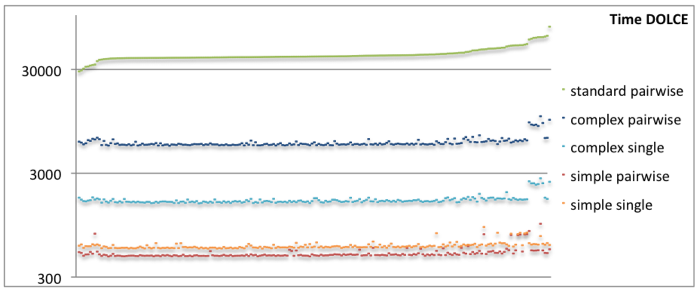

**Fig. 2.** The reasoning times in ms for testing the satisfiability of all concepts in DOLCE

**DOLCE** is a small but complex $\mathcal{SHOIQ}(\mathcal{D})$ ontology containing 209 concepts and 1,537 axioms that produce 2,325 HT-clauses. Core blocking works particularly well on DOLCE. The pre-model sizes (see Figure 1) and the reasoning times (see Figure 2) for all core blocking variants are consistently below those obtained with the standard anywhere blocking strategy. The simple single core blocking strategy gives the smallest pre-models but the reasoning times are slightly smaller for the simple pairwise strategy. This is because the simple single strategy produces more invalid blocks and, consequently, requires more expansion and (expensive) validation cycles before a final pre-model is found (see Table 1). Overall, the strategies work very well because DOLCE does not seem to be very highly constrained and many blocks are valid immediately.

**GALEN** is the original version of the GALEN medical ontology dating from about 10 years ago. Apart from CB [10], which implements an extension of an $\mathcal{EL}$-style algorithm to Horn-$\mathcal{SHIQ}$ [10], HermiT is currently the only reasoner that can classify this ontology. GALEN is a Horn-$\mathcal{SHIF}$ ontology containing 2,748 concepts and 4,979 axioms that produce 8,189 HT-clauses, and it normally requires pairwise blocking. GALEN is unusual in that it contains 2,256 “easy” concepts that are satisfied in very small pre-models ($<$ 200 individuals) and 492 “hard” concepts that are satisfied in very large pre-models ($>$ 35,000 individuals) for the standard blocking strategy. The classification times in Table 2 take all concepts into account; in all other cases we omit the measurements for the “easy” concepts since they do not show much difference between the different blocking strategies and just clutter the presentation. As for DOLCE, the

**Table 1.** Average measurements over all concepts in DOLCE and the classification time

| | standard pairwise | complex pairwise | complex single | simple pairwise | simple single |
|---|---|---|---|---|---|
| final pre-model size | 28,310 | 13,583 | 5,942 | 1,634 | 1,426 |
| finally blocked | 19,319 | 9,341 | 4,241 | 1,207 | 1,046 |
| number of validations | — | 1.03 | 1.06 | 1.09 | 2.09 |
| time in ms | 41,821 | 5,970 | 1,663 | 511 | 601 |
| validation part | — | 2.17% | 3.98% | 48.84% | 65.63% |
| classification time | 01:18:32 | 00:24:03 | 00:08:43 | 00:03:45 | 00:05:29 |

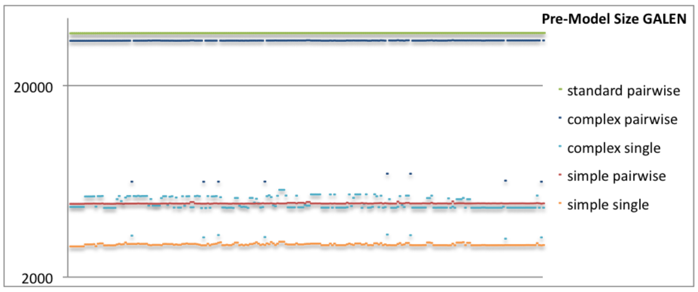

**Fig. 3.** The number of individuals in the pre-models for the (hard) concepts in GALEN

simple single core blocking strategy produces the most significant reduction in model size (see Figure 3). Although this strategy requires the most validation rounds, and these take up 87% of the overall reasoning time, this strategy is still the fastest (see Figure 4) since the reduction in model sizes compensates for the expensive block validations.

The only optimization in HermiT that needs adapting in order to work with core blocking is the *blocking cache*: once a pre-model for a concept is constructed, parts of the pre-model are reused in the remaining subsumption tests [4]. This dramatically reduces the overall classification time. The blocking cache can only be used on ontologies without nominals; in out test suite only GALEN falls into that category. Although the blocking cache could in principle be adapted for use with core blocking, this has not yet been implemented, so we switched this optimization off.

**The foundational model of anatomy (FMA)** is a domain ontology about human anatomy [2]. The ontology is one of the largest OWL ontologies available, containing $41,648$ concepts and $122,617$ $\mathcal{ALCOIF}(\mathcal{D})$ axioms, and it is transformed into $125,346$ HT-clauses and $3,740$ ABox assertions. For FMA, only the single simple core blocking strategy was able to process all concepts within the given one minute time limit per concept. The pairwise simple core strategy suffered from 229 timeouts, the single complex core strategy from $431$, the pairwise complex core strategy from $1,178$, and the standard blocking strategy from $1,253$. For the classification, we did not impose a timeout, but only the simple strategies managed to finish the classification with the 1GB memory limit. The simple pairwise strategy took about 25 hours, whereas the simple single strategy took 5.5 hours, discovering $33,431$ unsatisfiable concepts. The ontology thus seems to contain modeling errors that went undetected so far due to lack of

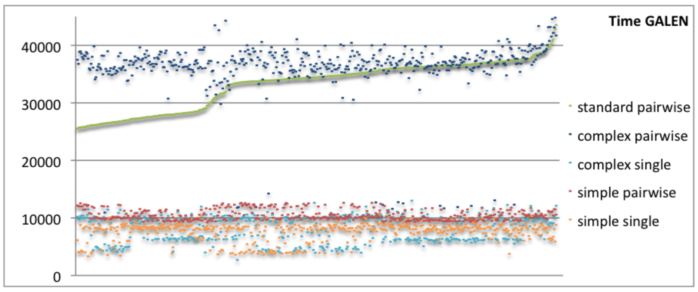

**Fig. 4.** The reasoning times for testing the satisfiability of the (hard) concepts in GALEN

**Table 2.** Average measurements over (hard) concepts in GALEN and the classification time

| | standard pairwise | complex pairwise | complex single | simple pairwise | simple single |
|---|---|---|---|---|---|
| final pre-model size | 37,747 | 33,557 | 4,876 | 4,869 | 2,975 |
| finally blocked | 19,290 | 19,726 | 2,234 | 1,896 | 1,247 |
| number of validations | — | 9.18 | 12.65 | 8.87 | 13.91 |
| time in ms | 33,293 | 36,213 | 8,050 | 10,485 | 7,608 |
| validation part | — | 26.28% | 76.67% | 81.59% | 87.58% |
| classification time | 03:50:01 | 04:35:12 | 01:07:18 | 01:27:50 | 01:02:44 |

adequate tool support. The unsatisfiability of all of these concepts was detected before blocking validation was required. Since only the simple single core blocking strategy succeeded in all tests, we only give complete measurements in Table 3 for this strategy. The sizes of the ABoxes constructed while processing unsatisfiable concepts is included in the final pre-model size, although these are not strictly pre-models since they contain a clash.

We also tested how much memory is necessary to construct all pre-models for DOLCE and GALEN under different blocking strategies. Starting with 16MB, we doubled the memory until the tested strategy could build all pre-models. The simple and complex core blocking strategies require as little as 64MB and 128MB of memory, respectively, whereas the standard blocking technique requires 512MB.

## 5 Discussion

In this paper we presented several novel blocking strategies that can improve the performance of DL reasoners by significantly reducing the size of the pre-models generated during satisfiability tests. Although we expected complex core blocking to work better on knowledge bases in expressive DLs, the evaluation shows that the simple core policy clearly outperforms the complex core policy regarding space and time on all tested ontologies. On more complex ontologies, the memory requirement with core blocking seems to decrease significantly.

**Table 3.** Average measurements over FMA with the single simple core strategy

| | | | |
|---|---|---|---|
| final pre-model size | 1,747 | finally blocked | 1,074 |
| time in ms | 518 | validation part | 0.00% |
| number of validations | 0.2 | classification time | 05:31:23 |

On ontologies where very few individuals are blocked (e.g., Wine) the new strategies cannot really reduce the sizes of the pre-models [3]; however, they do not seem to have a negative effect on the reasoning times either.

The current publicly available version of HermiT (1.2.2) uses simple single core blocking as its default blocking strategy for ontologies with nominals; for ontologies without nominals it uses standard anywhere blocking with the blocking cache optimization, an optimization that has not yet been extended to core blocking.

Blocking validation is not highly optimized in our prototypical implementation. This is most apparent for the single simple core strategy that causes the most invalid blocks and where block validation takes 86% of the time for GALEN. Only the significant model size reductions allows this strategy to nevertheless be the fastest. We believe that we can significantly improve the performance in the future. We identified the two most common reasons for invalid blocks: the $\geqslant$-rule is applicable to an assertion from $\mathsf{val}_{\mathcal{A}_\ell}(s)$ of a blocked individual, or the Hyp-rule is applicable to the assertions from the temporary ABox of the predecessor of a directly blocked individual. Testing for these two cases first should reduce the overall time of validity tests. Finally, we shall adapt the blocking cache technique to core blocking.

**Acknowledgements** The presented work is funded by the EPSRC project HermiT: Reasoning with Large Ontologies.

## References

1. Horrocks, I., Sattler, U.: Optimised reasoning for $\mathcal{SHIQ}$. In: Proc. of ECAI-02. (2002)
2. Golbreich, C., Zhang, S., Bodenreider, O.: The foundational model of anatomy in owl: Experience and perspectives. J. of Web Semantics: Science, Services and Agents on the World Wide Web **4**(3) (2006) 181–195
3. Glimm, B., Horrocks, I., Motik, B.: Optimized dl reasoning via core blocking. Technical report, University of Oxford (2010) http://www.comlab.ox.ac.uk/files/2743/paper.pdf.
4. Motik, B., Shearer, R., Horrocks, I.: Hypertableau reasoning for description logics. J. of Artificial Intelligence Research **173**(14) (2009) 1275–1309
5. Horrocks, I., Kutz, O., Sattler, U.: The even more irresistible $\mathcal{SROIQ}$. In: Proc. KR-06. (2006) 57–67
6. Baumgartner, P., Schmidt, R.A.: Blocking and Other Enhancements for Bottom-Up Model Generation Methods. In: Proc. of IJCAR-06. Volume 4130 of LNCS. (2006) 125–139
7. Ding, Y., Haarslev, V.: Tableau Caching for Description Logics with Inverse and Transitive Roles. In: Proc. DL. (2006)
8. Goré, R., Widmann, F.: Sound global state caching for $\mathcal{ALC}$ with inverse roles. In: Proc. of TABLEAUX 2009. LNCS (2009) 205–219
9. Donini, F.M., Massacci, F.: EXPTIME tableaux for $\mathcal{ALC}$. Artificial Intelligence Journal **124**(1) (2000) 87–138
10. Kazakov, Y.: Consequence-driven reasoning for horn SHIQ ontologies. In: Proc. of IJCAI-09. (2009) 2040–2045