Individual Reuse in Description Logic Reasoning

Boris Motik and Ian Horrocks

University of Oxford, UK

Abstract. Tableau calculi are the state-of-the-art for reasoning in description logics (DL). Despite recent improvements, tableau-based reasoners still cannot process certain knowledge bases (KBs), mainly because they end up building very large models. To address this, we propose a tableau calculus with *individual reuse*: to satisfy an existential assertion, our calculus nondeterministically tries to reuse individuals from the model generated thus far. We present two *expansion strategies*: one is applicable to the DL $\mathcal{ELHO}$ and gives us a worst-case optimal algorithm, and the other is applicable to the DL $\mathcal{SHOIQ}$. Using this technique, our reasoner can process several KBs that no other reasoner can.

1 Introduction

Description Logics (DLs) [2] are used for conceptual modeling in diverse areas of computer science. This is largely due to the practical support for automated reasoning, which can help users during modeling. Practical DL reasoners are mostly based on tableau calculi [2], which are essentially model building algorithms. Tableau calculi, as well as the underlying computational problems, are of high computational complexity, so various optimizations of the basic algorithm have been developed [2, Chapter 9] and incorporated into reasoners such as FaCT++ [16], Pellet [12], and RACER [8].

DLs are often used in life sciences, and this continuously poses new challenges for DL research. For example, GALEN [13] and FMA [14]—detailed and comprehensive models of human anatomy—have both been translated into DLs, but the resulting knowledge bases cannot be processed using existing DL reasoners. To address this problem we recently proposed a novel DL reasoning algorithm [10, 11] based on *hypertableau* [4]. Unlike the standard tableau calculi, the hypertableau calculus is deterministic if a knowledge base can be translated into a Horn theory; furthermore, it uses *anywhere blocking*, which can significantly reduce the sizes of the generated models. Our HermiT reasoner, which implements the new calculus, was the first one to successfully process the original version of GALEN—that is, the version from ten years ago.

Despite these improvements, many DL knowledge bases are still out of HermiT's reach. The hypertableau calculus minimizes nondeterminism, so the remaining performance problems are due to the construction of very large models. We therefore propose an extension to the hypertableau calculus based on *individual reuse*: to satisfy an existential assertion $(\exists R.C)(s)$, our calculus first tries to *reuse* an individual from the model constructed thus far; if this fails, the calculus

then introduces a fresh individual. We focus here on the hypertableau calculus; however, individual reuse could be applied in standard tableau calculi as well. In Section 3, we discuss the practical rationale behind our approach. In particular, we observe that, due to certain restrictions of DL languages, DL knowledge bases are often underconstrained, thus naturally allowing for individual reuse.

If applied naïvely, individual reuse would yield a highly nondeterministic calculus. We address this problem in two steps. First, we encapsulate the part of the calculus that determines which individuals to reuse in an *expansion strategy*, and we identify general conditions that a strategy must satisfy in order for the algorithm to be sound, complete, and terminating.[1] Second, we present two particular strategies. For knowledge bases expressed in $\mathcal{ELHO}$—a fragment of the tractable DL $\mathcal{EL}$++ [1]—we show that individual reuse can be done *deterministically*, yielding a polynomial algorithm, and we identify a close correspondence between our calculus and the $\mathcal{EL}$++ algorithm from [1]. For knowledge bases expressed in $\mathcal{SHOIQ}$—the DL underlying the Semantic Web ontology language OWL—we present an expansion strategy that generalizes the $\mathcal{ELHO}$ one.

We have extended HermiT with individual reuse and have conducted a preliminary evaluation on practical ontologies, of which several are used in life sciences. Unlike FaCT++ and Pellet, HermiT can classify the BAMS ontology[2] and a particular fragment of FMA. Furthermore, HermiT can solve individual classification tests on a "hard" fragment of the current version of GALEN; however, each test takes about 40 seconds, which makes classifying the knowledge base infeasible given the large number of concepts. The new algorithm also improves HermiT's performance on complex ontologies such as DOLCE. Thus, individual reuse seems to promise significant improvements in practical DL reasoning.

Please refer to [11] for nonessential technical details. All proofs are given in Appendix A.

2 Preliminaries

The description logic $\mathcal{SHOIQ}$ is defined as follows. A $\mathcal{SHOIQ}$ *signature* is a triple $\Sigma = (N_R, N_C, N_I)$ consisting of disjoint sets of *atomic roles* N_R, *atomic concepts* N_C, and *individuals* N_I. The set of *roles* is $N_R \cup \{R^- \mid R \in N_R\}$. For $R \in N_R$, let $\mathsf{Inv}(R) = R^-$ and $\mathsf{Inv}(R^-) = R$. An *RBox* $\mathcal{R}$ is a finite set of *role inclusions* $R \sqsubseteq S$ and *transitivity axioms* $\mathsf{Trans}(R)$ for R and S roles. Let $\sqsubseteq^*_{\mathcal{R}}$ be the reflexive-transitive closure of $\{R \sqsubseteq S, \mathsf{Inv}(R) \sqsubseteq \mathsf{Inv}(S) \mid R \sqsubseteq S \in \mathcal{R}\}$. A role R is *transitive* in $\mathcal{R}$ if a role S exists such that $S \sqsubseteq^*_{\mathcal{R}} R$, $R \sqsubseteq^*_{\mathcal{R}} S$, and either $\mathsf{Trans}(S) \in \mathcal{R}$ or $\mathsf{Trans}(\mathsf{Inv}(S)) \in \mathcal{R}$; furthermore, R is *simple* if no transitive role S exists such that $S \sqsubseteq^*_{\mathcal{R}} R$. The set of *concepts* is the smallest set containing concepts shown in the left-hand side of Table 1, for $A \in N_C$, $a \in N_I$, C and D concepts, R a role, S a simple role, and n a nonnegative integer; concepts of the form $\{a\}$, $\geq n\, S.C$, and $\leq n\, S.C$ are called *nominals*, *at-least*, and *at-most*

[1] We use "soundness" and "completeness" as in resolution theorem proving: a sound calculus preserves satisfiability; a complete calculus correctly detects satisfiability.

[2] `http://brancusi.usc.edu/bkms/`

Table 1: Model-Theoretic Semantics of $\mathcal{SHOIQ}$

Semantics of Roles and Concepts		Semantics of Axioms	
$\top^I = \triangle^I$	$\bot^I = \emptyset$	$C \sqsubseteq D$	$\Rightarrow C^I \subseteq D^I$
$\{s\}^I = \{s^I\}$	$(\neg C)^I = \triangle^I \setminus C^I$	$R \sqsubseteq S$	$\Rightarrow R^I \subseteq S^I$
$(C \sqcap D)^I = C^I \cap D^I$	$(C \sqcup D)^I = C^I \cup D^I$	$\mathsf{Trans}(R)$	$\Rightarrow (R^I)^+ \subseteq R^I$
$(R^-)^I = \{\langle b, a\rangle \mid \langle a, b\rangle \in R^I\}$		$C(a)$	$\Rightarrow a^I \in C^I$
$(\forall R.C)^I = \{x \mid \forall y : \langle x, y\rangle \in R^I \rightarrow y \in C^I\}$		$R(a, b)$	$\Rightarrow \langle a^I, b^I\rangle \in R^I$
$(\exists R.C)^I = \{x \mid \exists y : \langle x, y\rangle \in R^I \wedge y \in C^I\}$		$a \approx b$	$\Rightarrow a^I = b^I$
$(\leq n\, S.C)^I = \{x \mid \sharp\{y \mid \langle x, y\rangle \in S^I \wedge y \in C^I\} \leq n\}$		$a \not\approx b$	$\Rightarrow a^I \neq b^I$
$(\geq n\, S.C)^I = \{x \mid \sharp\{y \mid \langle x, y\rangle \in S^I \wedge y \in C^I\} \geq n\}$			

Note: $\sharp N$ is the number of elements in N, and R^+ is the transitive closure of R.

concepts, respectively. A *TBox* $\mathcal{T}$ is a finite set of *general concept inclusions* (GCIs) $C \sqsubseteq D$ for C and D concepts. An *ABox* $\mathcal{A}$ is a finite set of assertions of the form $C(a)$, $R(a, b)$, and (in)equalities $a \approx b$ and $a \not\approx b$, for C a concept, R a role, and a and b individuals. A $\mathcal{SHOIQ}$ *knowledge base* $\mathcal{K}$ is a triple $(\mathcal{R}, \mathcal{T}, \mathcal{A})$.

An *interpretation* for $\mathcal{K}$ is a tuple $I = (\triangle^I, \cdot^I)$, where $\triangle^I$ is a nonempty set, and $\cdot^I$ assigns an element $a^I \in \triangle^I$ to each individual a, a set $A^I \subseteq \triangle^I$ to each atomic concept A, and a relation $R^I \subseteq \triangle^I \times \triangle^I$ to each atomic role R. The function $\cdot^I$ is extended to concepts and roles as shown in the left-hand side of Table 1. I is a *model* of $\mathcal{K}$, written $I \models \mathcal{K}$, if it satisfies all axioms of $\mathcal{K}$ as shown in the right-hand side of Table 1. The basic inference problem for $\mathcal{SHOIQ}$ is checking *satisfiability* of $\mathcal{K}$—that is, checking whether a model of $\mathcal{K}$ exists.

The DL $\mathcal{ELOH}$ [3] is obtained from $\mathcal{SHOIQ}$ by disallowing transitive and inverse roles, and by allowing only concepts of the form $\top$, $\bot$, A, $\{a\}$, $\exists R.C$, and $C_1 \sqcap C_2$. The DL $\mathcal{SHIQ}$ is obtained from $\mathcal{SHOIQ}$ by disallowing nominals.

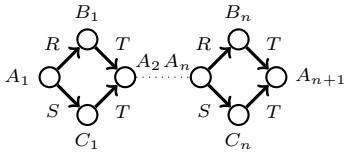
3 Motivation

For a TBox $\mathcal{T}$ and an ABox $\mathcal{A}$, a tableau calculus evolves $\mathcal{A}$ towards a representation of a model by applying derivation rules. The $\exists$-rule is found in virtually all tableau calculi: given $(\exists R.C)(s) \in \mathcal{A}_i$, it introduces a fresh individual t and derives the ABox $\mathcal{A}_{i+1} := \mathcal{A}_i \cup \{R(s, t), C(t)\}$. The rule is applied only if s is not *blocked* in $\mathcal{A}_i$; roughly speaking, this is the case if no individual u exists such that $D(s) \in \mathcal{A}_i$ if and only if $D(u) \in \mathcal{A}_i$ for each concept D. Apart from ensuring termination, blocking significantly reduces model sizes in practice [10].

DLs usually enjoy a *tree model property*: each satisfiable knowledge base has a tree-shaped model. While this is useful for ensuring decidability [17], it also prevents us from fully axiomatizing nontree structures. For example, a structure such as the one shown in Figure 2a can only be approximated using the axioms shown in Figure 2b. The axioms in Figure 2b have a model I corresponding to Figure 2a; however, they also have an exponentially larger model I' obtained by "unfolding" Figure 2a into a tree.

Consider now a run of our hypertableau algorithm on a knowledge base $\mathcal{K}$ containing the axioms in Figure 2b and the axioms $A_{n+1} \sqsubseteq E$, $\exists T.E \sqsubseteq E$, and

Fig. 1: Problems with Model Construction



(a) The Intended Structure

$$A_1 \sqsubseteq \exists R.B_1 \sqcap \exists S.C_1 \qquad A_n \sqsubseteq \exists R.B_n \sqcap \exists S.C_n$$
$$B_1 \sqsubseteq \exists T.A_2 \qquad \ldots \quad B_n \sqsubseteq \exists T.A_{n+1}$$
$$C_1 \sqsubseteq \exists T.A_2 \qquad C_n \sqsubseteq \exists T.A_{n+1}$$

(b) The Axiomatization in DLs

$\exists R.E \sqcap \exists S.E \sqsubseteq E$. The algorithm exploits the tree model property by constructing only (representations of) tree-shaped models. It will thus initially construct a fragment of the exponential tree: the fragment will, for example, contain two individuals labeled with A_2 such that one blocks the other. But then E will be added to all existing individuals, which will invalidate blocking. Thus, our algorithm eventually constructs the entire exponential tree. Different blocking and rule application strategies are known that might prevent the generation of the exponential tree in this example; however, the example can be modified such that the exponential tree is generated in spite of these optimizations.

Complex structures abound in life science ontologies; for example, GALEN states that "the left ventricle is a solid division of the left side of the heart," "the left ventricle is a beta connection of the mitral valve," "the mitral valve is a structural component of the left side of the heart," and so on. The intended structure is much more complex than the one shown Figure 2a, and the axioms are cyclic, so tableau reasoners generate very large tree models. It is reasonable, however, to expect that GALEN is satisfied in a relatively small non-tree-shaped model that contains only one left side, one left ventricle, and one mitral valve.

We use these observations in our new calculus: given $(\exists R.C)(s)$, instead of always creating a fresh individual, our calculus first tries to reuse an individual from the model generated thus far. Our calculus is similar to the tableau calculus for first-order logic from [5]; however, the latter calculus is not a decision procedure for DLs that lack the finite-model property and is unlikely to be suitable for practice due to its very large degree of nondeterminism.

4 The Hypertableau Algorithm with Individual Reuse

Our calculus with individual reuse is based on the hypertableau calculi for $\mathcal{SHIQ}$ [10] and $\mathcal{SHOIQ}$ [11]. Therefore, we first present an informal overview of these calculi in Section 4.1, and then formally introduce the calculus with individual reuse in Section 4.2.

4.1 The Standard Hypertableau Calculus for $\mathcal{SHOIQ}$

Our hypertableau calculus is related to hyperresolution with splitting, which has been used to obtain a decision procedure for several description and modal logics [9, 6]. The algorithm consists of the preprocessing and the hypertableau phases.

The *preprocessing phase* translates a $\mathcal{SHOIQ}$ knowledge base $\mathcal{K}$ into a *normalized* ABox $\Xi_{\mathcal{A}}(\mathcal{A})$ and a set $\Xi_{\mathcal{TR}}(\mathcal{K})$ of *DL-clauses*—universally quantified first-order implications of the form $\bigwedge U_i \rightarrow \bigvee V_j$ where U_i and V_j are *atoms* of the form $R(x, y)$, $C(x)$, and $x \approx y$, for x and y variables, R an atomic role, C a concept, and $\approx$ the equality predicate. This transformation is an optimized version of the well-known structural transformation followed by a translation of certain concepts into first-order logic; please refer to [11, Section 4.1] for details. The translation produces *HT-clauses*—syntactically restricted DL-clauses on which our hypertableau calculus is guaranteed to terminate. A precise definition of HT-clauses is given in [11, Definition 7]. Roughly speaking, HT-clauses can have the form (1), where R_i and S_i are (not necessarily atomic) roles, A_i and B_i are atomic concepts, and C_i and D_i are either atomic or concepts of the form $\geq n\, R.A$ or $\geq n\, R.\neg A$. Furthermore, ar is a function defined as $\mathsf{ar}(R, s, t) = R(s, t)$ and $\mathsf{ar}(R^-, s, t) = R(t, s)$ for R an atomic role and s and t individuals or variables.

$$\begin{aligned}(1)\qquad &\bigwedge A_i(x) \wedge \bigwedge \mathsf{ar}(R_i, x, y_i) \wedge \bigwedge B_i(y_i) \wedge \bigwedge O_{a_i}(y_{a_i}) \rightarrow \\ &\qquad \bigvee C_i(x) \vee \bigvee D_i(y_i) \vee \bigvee \mathsf{ar}(S_i, x, y_i) \vee \bigvee x \approx y_{a_i} \vee \bigvee y_i \approx y_j \; @^x_{\leq n\, R.C}\end{aligned}$$

The atoms of the form $x \approx y_{a_i}$ stem from nominals; for example, $C \sqsubseteq \{a\}$ is translated into an HT-clause $C(x) \wedge O_a(y_a) \rightarrow x \approx y_a$ and an assertion $O_a(a)$. A concept O_a is uniquely associated with each nominal $\{a\}$ and it is called a *nominal guard concept*; such concepts are used to "push" all individuals from DL-clauses into the ABox. Finally, the *at-most equalities* $y_i \approx y_j \; @^x_{\leq n\, R.C}$ stem from the translation of at-most concepts; for example, $\top \sqsubseteq \leq 1\, R.\top$ is translated into $R(x, y_1) \wedge R(x, y_2) \rightarrow y_1 \approx y_2 \; @^x_{\leq 1\, R.\top}$. The *annotation* $@^x_{\leq 1\, R.\top}$ does not affect the meaning of the equality; it merely records its provenance, and we shall discuss the usage of this provenance information shortly. The concept $\exists R.C$ is used in the rest of this paper as an abbreviation for $\geq 1\, R.C$.

The *hypertableau phase* decides satisfiability of a set of HT-clauses $\mathcal{C}$ and an ABox $\mathcal{A}$. The main derivation rule is similar to the one of the hypertableau calculus for first order logic [4]: given an HT-clause $\bigwedge_{i=1}^{m} U_i \rightarrow \bigvee_{j=1}^{n} V_j$ and an ABox $\mathcal{A}$, the *Hyp*-rule tries to unify the atoms $U_1, \ldots, U_m$ with a subset of the assertions in $\mathcal{A}$; if a unifier σ is found, the rule nondeterministically derives $\sigma(V_j)$ for some $1 \leq j \leq n$. For example, given $R(x, y) \rightarrow \exists R.C(x) \vee D(y)$ and an assertion $R(a, b)$, the *Hyp*-rule derives either $\exists R.C(a)$ or $D(b)$. The $\geq$-rule deals with existential quantifiers: given $\exists R.C(a)$, the rule introduces a fresh individual t and derives $R(a, t)$ and $C(t)$. The $\approx$-rule deals with equality: given $a \approx b$, the rule replaces the individual a in all assertions with the individual b, and it introduces a *renaming* $a \mapsto b$ in order to keep track of the merging. We take $\approx$ to have built-in symmetry; thus, $a \approx b$ should also be read as $b \approx a$. Finally, the $\bot$-rule detects contradictions such as $A(a)$ and $\neg A(a)$, or $a \not\approx a$.

Termination of the calculus is ensured through *blocking*, which is based on the key concept of *forest-shaped ABoxes*. We discuss here just the intuition behind the concept; please refer to [11, Definition 11] for details. A forest-shaped ABox is shown in Figure 2, where nodes and edges correspond to individuals and role assertions, respectively. Individuals in such an ABox can be separated into two

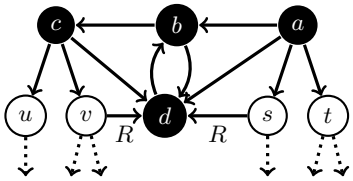


Fig. 2: Forest-Shaped ABoxes

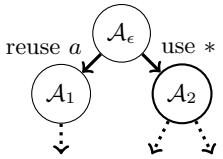


Fig. 3: Example Derivation

sets. *Named* individuals (shown as black nodes) originate from the *input ABox*, and they can be connected in arbitrary ways. *Blockable* individuals (shown as white nodes) are introduced by the $\geq$-rule, and they can be connected either to arbitrary named individuals, or to other blockable individuals in a *tree-like* way. Blockable individuals are represented as strings; for example, $s = a.1$ denotes that s is the first *successor* of a. These notions can be used to identify certain individuals as *blocked*. The $\geq$-rule is applied only to nonblocked individuals, and this ensures termination without affecting completeness.

As shown in [11, Lemma 12], applications of most derivation rules preserve the forest shape of an ABox; however, inverse roles, nominals, and number restrictions cause subtle problems. Consider again Figure 2 and assume that d must satisfy an at-most restriction $\leq 1\, R^-.\top$. This implies $v \approx s$, so one individual should be merged into the other; however, this can compromise the tree shape of the ABox. The *NI*-rule deals with this problem by promoting one of v or s into a *root individual*: such individuals can be connected in arbitrary ways even if they do not occur in the input ABox. Thus, an application of the *Hyp*-rule to the ABox in Figure 2 and the HT-clause $R(y_1, x) \wedge R(y_2, x) \rightarrow y_1 \approx y_2\, @^x_{\leq 1\, R^-.\top}$ derives the at-most equality $v \approx s\, @^d_{\leq 1\, R^-.\top}$. By examining the annotation on the equality, the *NI*-rule can see that the equality stems from an at-most concept, so it turns either v or s into a root individual. It is possible to establish a bound on the number of the introduced root individuals and thus ensure termination.

4.2 Introducing Individual Reuse

In this section, we extend the hypertableau calculus with individual reuse.

Definition 1 (Hypertableau with Individual Reuse).

Individuals. *Given a set of* named *individuals N_I, the set of* root individuals *N_O is the smallest set such that $N_I \subseteq N_O$ and, if $x \in N_O$, then $x.\langle R, B, i\rangle \in N_O$ for each role R, each integer i, and each B of the form A or $\neg A$ with A an atomic concept. The set of* all individuals *N_A is the smallest set such that $N_O \subseteq N_A$ and, if $x \in N_A$, then $x.i \in N_A$ for each integer i. The individuals in $N_A \setminus N_O$ are* blockable individuals*. A blockable individual $x.i$ is a* successor *of x, and x is a* predecessor *of $x.i$.* Descendant *and* ancestor *are the transitive closures of successor and predecessor, respectively.*

ABoxes. *An ABox that contains only named individuals and no at-most equalities is called an* input ABox*. The hypertableau algorithm works with* generalized ABoxes*, which can contain assertions using the individuals from N_A, a*

special assertion $\bot$ *that is false in all interpretations, and an acyclic and confluent relation* $\mapsto$ *on root individuals called* renaming. *The* canonical name *of a root individual* $a \in N_O$ *w.r.t.* $\mathcal{A}$*, written* $\|a\|_\mathcal{A}$*, is the normal form of a w.r.t.* $\mapsto$ *in* $\mathcal{A}$*. If* a *occurs in* $\mathcal{A}$*, then the relation* $\mapsto$ *must be such that* $\|a\|_\mathcal{A} = a$*.*

Pairwise Anywhere Blocking. *The* label of an individual *is defined as* $\mathcal{L}_\mathcal{A}(s) = \{C \mid C(s) \in \mathcal{A}$ *and* C *is of the form* $A, \geq n\, R.A$ *or* $\geq n\, R.\neg A\}$*, and* of an individual pair *as* $\mathcal{L}_\mathcal{A}(s,t) = \{R \mid R(s,t) \in \mathcal{A}\}$*. Let* $\prec$ *be a transitive and irreflexive relation on* N_A *such that, if* s' *is an ancestor of* s*, then* $s' \prec s$*. By induction on* $\prec$*, we assign to each individual in* $\mathcal{A}$ *a status as follows: a blockable individual* s *with a predecessor* s' *is* directly blocked by *a blockable individual* t *with a predecessor* t' *iff* t *is not blocked,* $t \prec s$*,* $\mathcal{L}_\mathcal{A}(s) = \mathcal{L}_\mathcal{A}(t)$*,* $\mathcal{L}_\mathcal{A}(s') = \mathcal{L}_\mathcal{A}(t')$*,* $\mathcal{L}_\mathcal{A}(s,s') = \mathcal{L}_\mathcal{A}(t,t')$*, and* $\mathcal{L}_\mathcal{A}(s',s) = \mathcal{L}_\mathcal{A}(t',t)$*;* s *is* indirectly blocked *iff it has a predecessor that is blocked; and* s *is* blocked *iff it is directly or indirectly blocked.*

Pruning. *The ABox* $\mathsf{prune}_\mathcal{A}(s)$ *is obtained from* $\mathcal{A}$ *by removing all assertions containing a descendent of* s*.*

Merging. *The ABox* $\mathsf{merge}_\mathcal{A}(s \to t)$ *is obtained from* $\mathsf{prune}_\mathcal{A}(s)$ *by replacing the individual* s *with the individual* t *in all assertions (but not in the renaming relation* $\mapsto$*) and, if both* s *and* t *are root individuals, adding the renaming* $s \mapsto t$*.*

Expansion Strategy. *An* expansion strategy *is a function that, for each concept* $\geq n\, R.C$*, individual* s*, and ABox* $\mathcal{A}$*, returns a* k*-tuple of* n*-tuples of the form* $\mathsf{iexp}(\geq n\, R.C, s, \mathcal{A}) = [(\gamma_{1,1}, \ldots, \gamma_{1,n}), \ldots, (\gamma_{k,1}, \ldots, \gamma_{k,n})]$*, where* $k \geq 1$ *and, for each* $1 \leq i \leq k$ *and* $1 \leq j \leq n$*,* $\gamma_{i,j}$ *is a special symbol* $*$*, a predecessor or a successor of* s*, or a root individual (which may or may not occur in* $\mathcal{A}$*).*

Derivation Rules. *Table 2 specifies rules that, for* $\mathcal{A}$ *an ABox,* $\mathcal{C}$ *a set of HT-clauses, and* iexp *an expansion strategy, derive the ABoxes* $\mathcal{A}_1, \ldots, \mathcal{A}_\ell$*.*

Rule Precedence. *The* $\approx$*-rule can be applied to a (possibly annotated) equality* $s \approx t$ *in an ABox* $\mathcal{A}$ *only if* $\mathcal{A}$ *does not contain an equality of the form* $s \approx t\, @^u_{\leq n\, R.B}$ *to which the NI-rule is applicable.*

Clash. *An ABox* $\mathcal{A}$ *contains a* clash *iff* $\bot \in \mathcal{A}$*; otherwise,* $\mathcal{A}$ *is* clash-free*.*

Derivation. *A* derivation $D = (T, \lambda)$ *for a set of HT-clauses* $\mathcal{C}$*, an ABox* $\mathcal{A}$*, and an expansion strategy* iexp *consists of a finitely branching tree* T *and a function* λ *labeling the nodes of* T *with ABoxes such that (*i*)* $\lambda(\epsilon) = \mathcal{A}$ *for* ϵ *the root of* T*, (*ii*)* $t \in T$ *is a leaf of* T *if* $\bot \in \lambda(t)$ *or no derivation rule is applicable to* $\lambda(t)$ *and* $\mathcal{C}$*, and (*iii*) otherwise,* $t \in T$ *has children* $t_1, \ldots, t_\ell$ *such that* $\lambda(t_1), \ldots, \lambda(t_\ell)$ *are exactly the results of applying one (arbitrarily chosen, but respecting the precedence) applicable derivation rule to* $\lambda(t)$ *and* $\mathcal{C}$*. The derivation* D *is* successful *if it contains a leaf node labeled with a clash-free ABox.*

The main difference to the hypertableau calculus from [11, Definition 10] is in the $\geq$-rule and the notion of an *expansion strategy*. Intuitively, given an assertion $\geq n\, R.C(s) \in \mathcal{A}$, the new $\geq$-rule consults the expansion strategy iexp to determine the possible ways of satisfying the assertion. The strategy can identify k different ways to do this; if $k > 1$, the $\geq$-rule becomes nondeterministic. The i-th variant is described as an n-tuple $(\gamma_{i,1}, \ldots, \gamma_{i,n})$, in which $\gamma_{i,j}$ specifies how to obtain the j-th of the n required individuals. For each $\gamma_{i,j}$, the strategy can cause the $\geq$-rule to either (i) reuse an existing individual from $\mathcal{A}$, (ii) introduce

Table 2: Derivation Rules of the Tableau Calculus

Hyp-rule	If 1. $U_1 \wedge \ldots \wedge U_m \to V_1 \vee \ldots \vee V_n \in \mathcal{C}$, and 2. a mapping σ of variables to the individuals of $\mathcal{A}$ exists such that 2.1 $\sigma(x)$ is not indirectly blocked for each variable $x \in N_V$, 2.2 $\sigma(U_i) \in \mathcal{A}$ for each $1 \leq i \leq m$, and 2.3 $\sigma(V_j) \notin \mathcal{A}$ for each $1 \leq j \leq n$, then $\mathcal{A}_1 := \mathcal{A} \cup \{\bot\}$ if $n = 0$; $\mathcal{A}_j := \mathcal{A} \cup \{\sigma(V_j)\}$ for $1 \leq j \leq n$ otherwise.
$\geq$-rule	If 1. $\geq n\, R.C(s) \in \mathcal{A}$, 2. s is not blocked in $\mathcal{A}$, 3. $\mathcal{A}$ does not contain individuals $u_1, \ldots, u_n$ such that 3.1 $\{\mathsf{ar}(R, s, u_i), C(u_i) \mid 1 \leq i \leq n\} \cup \{u_i \not\approx u_j \mid 1 \leq i < j \leq n\} \subseteq \mathcal{A}$, and 3.2 either s is blockable or no u_i, $1 \leq i \leq n$, is indirectly blocked in $\mathcal{A}$, and 4. $\mathsf{iexp}(\geq n\, R.C, s, \mathcal{A}) = [(\gamma_{1,1}, \ldots, \gamma_{1,n}), \ldots, (\gamma_{k,1}, \ldots, \gamma_{k,n})]$ then for each $1 \leq i \leq k$, $\mathcal{A}_i := \mathcal{A} \cup \{\mathsf{ar}(R, s, t_{i,j}), C(t_{i,j}) \mid 1 \leq j \leq n\} \cup \{t_{i,j} \not\approx t_{i,k} \mid 1 \leq j < k \leq n\}$ where $t_{i,j}$ is a fresh successor of s if $\gamma_{i,j} = *$, and $t_{i,j} = \gamma_{i,j}$ if $\gamma_{i,j} \neq *$.
$\approx$-rule	If 1. $s \approx t \in \mathcal{A}$ (the equality can possibly be annotated), and 2. $s \neq t$ then $\mathcal{A}_1 := \mathsf{merge}_{\mathcal{A}}(s \to t)$ if t is a named individual, or t is a root individual and s is not a named individual, or s is a descendant of t; $\mathcal{A}_1 := \mathsf{merge}_{\mathcal{A}}(t \to s)$ otherwise.
$\bot$-rule	If $s \not\approx s \in \mathcal{A}$ or $\{A(s), \neg A(s)\} \subseteq \mathcal{A}$ then $\mathcal{A}_1 := \mathcal{A} \cup \{\bot\}$.
NI-rule	If 1. $s \approx t\, @^u_{\leq n\, R.B} \in \mathcal{A}$ or $t \approx s\, @^u_{\leq n\, R.B} \in \mathcal{A}$, 2. u is a root individual, 3. s is a blockable individual and it is not a successor of u, and 4. t is a blockable individual then $\mathcal{A}_i := \mathsf{merge}_{\mathcal{A}}(s \to \|u.\langle R, B, i\rangle\|_{\mathcal{A}})$ for each $1 \leq i \leq n$.

a fresh root individual, or (*iii*) resort to the standard tableau behavior and introduce a fresh distinct blockable successor of s (when $\gamma_{i,j} = *$).

To preserve the forest shape of ABoxes, iexp cannot reuse an arbitrary individual from $\mathcal{A}$. For example, if the strategy reused u when expanding $\exists R.C(s)$ in the ABox shown in Figure 2, the resulting ABox would not be tree-shaped. Therefore, if $\gamma_{i,j} \neq *$, then $\gamma_{i,j}$ must be either a root individual, a predecessor of s, or a successor of s, thus ensuring that the resulting ABox is tree-shaped. A typical expansion strategy will use this definition as follows: the strategy will first introduce fresh root individuals and designate them as possible "reuse targets"; in subsequent inferences, the strategy will try to reuse these targets; finally, if such reuse fails, the strategy will resort to the standard behavior.

A strategy can thus introduce an unbounded number of root individuals; since these do not participate in blocking, the calculus is not guaranteed to terminate. Therefore, we require a strategy to be *bounded*—that is, to introduce a finite number of fresh individuals in each possible derivation.

Definition 2 (Bounded Strategy). *A strategy* iexp *is* bounded *if the number of individuals $\gamma_{i,j}$ such that $\gamma_{i,j}$ occurs in* $\mathsf{iexp}(\geq n\, R.C, s, \mathcal{A})$ *but not in $\mathcal{A}$ for some concept $\geq n\, R.C$, individual $s \in N_O$, and ABox $\mathcal{A}$ is finite.*

Since individual reuse preserves the forest shape of ABoxes, given a clash-free ABox $\mathcal{A}$ to which no derivation rule is applicable, we can construct a model for $\mathcal{A}$ just like in [11, Lemma 14]. In contrast, individual reuse is *unsound*: an application of the $\geq$-rule to a satisfiable ABox can result in an unsatisfiable ABox. The following definition introduces derivations in which reusing individuals does not lead to unsoundness.

Definition 3 (Safe Derivation). *Let $D = (T, \lambda)$ be a derivation for $\mathcal{C}$, $\mathcal{A}$, and* iexp*. The set of* safe *nodes of T is inductively defined such that $\epsilon \in T$ is safe, and $t.i \in T$ is safe if t is safe and (*i*) $\lambda(t.i)$ has not been obtained from $\lambda(t)$ by the $\geq$-rule, or (*ii*) $\lambda(t.i)$ has been obtained by applying the $\geq$-rule to $\geq n\, R.C(s) \in \lambda(t)$, and $(\gamma_{i,1}, \ldots, \gamma_{i,n})$ is the i-th n-tuple of* $\mathsf{iexp}(\geq n\, R.C, s, \mathcal{A})$ *such that, for each $1 \leq j \leq n$, either $\gamma_{i,j} = *$ or $\gamma_{i,j}$ is a named individual not occurring in $\lambda(t)$. The derivation D is* safe *if every nonleaf safe node has at least one safe child.*

Definition 3 might seem unnecessarily complex: soundness is ensured if we require each $\mathsf{iexp}(\geq n\, R.C, s, \mathcal{A})$ to contain a tuple $(*, \ldots, *)$, thus always allowing for the standard existential expansion. This is, however, unnecessarily restrictive. Consider the derivation shown in Figure 3, in which the $\geq$-rule is applied to $\mathcal{A}_\epsilon$, deriving $\mathcal{A}_1$ via individual reuse and $\mathcal{A}_2$ using the standard expansion. The first derivation is possibly unsound, so the second derivation is necessary in order to guarantee soundness; but then, there is no need to enforce soundness in the inferences below $\mathcal{A}_1$: applications of the $\geq$-rule below $\mathcal{A}_1$ can be performed in a way that can, but does not need to include $(*, \ldots, *)$ in $\mathsf{iexp}(\geq n\, R.C, s, \mathcal{A})$. Definition 3 formalizes this intuition: starting from the root ϵ, each derivation node that is obtained via a sound sequence of inferences must have at least one child obtained by a sound inference. The strategy that we present in Section 5.2 uses this definition: the first time a "reuse target" t is introduced, this is done nondeterministically; however, all subsequent reuses of t are deterministic. In other words, once our strategy nondeterministically decides to reuse t, it stays committed to this choice. We now present the main result of this section.

Theorem 1. *For $\mathcal{C}$ a set of HT-clauses, $\mathcal{A}$ an input ABox, and* iexp *a bounded expansion strategy, (1) each derivation for $\mathcal{C}$, $\mathcal{A}$, and* iexp *is finite; (2) if a successful derivation for $\mathcal{C}$, $\mathcal{A}$, and* iexp *exists, then $(\mathcal{C}, \mathcal{A})$ is satisfiable; and (3) if $(\mathcal{C}, \mathcal{A})$ is satisfiable, then each safe derivation for $\mathcal{C}$, $\mathcal{A}$, and* iexp *is successful.*

As discussed in [11], the *NI*-rule is not needed on HT-clauses obtained from $\mathcal{SHIQ}$ knowledge bases. With individual reuse, however, this is not the case: reusing a root individual c when expanding $\exists R.C(s)$ in Figure 2 can clearly trigger the *NI*-rule if c must satisfy an at-most restriction on R^-.

The *NI*-rule can introduce a performance penalty in practice, so we identify the class of *NI*-free strategies on which the rule is not needed even with individual reuse. Intuitively, an *NI*-free strategy can satisfy existential restrictions

on root individuals either by introducing fresh root individuals or by reusing existing ones; however, it always applies the standard expansion for blockable individuals. For example, such a strategy might at first introduce and reuse only root individuals; if this proves ineffective and the models get large, it might then decide not to reuse individuals in further applications of the $\geq$-rule.

Definition 4 (*NI*-free Strategies). *An expansion strategy* iexp *is NI-*free *if* $\mathsf{iexp}(\geq n\, R.C, s, \mathcal{A}) = [(*, \ldots, *)]$ *whenever* s *is a blockable individual.*

Proposition 1. *In a derivation where* $\mathcal{C}$ *is a set of HT-clauses not containing equalities of the form* $x \approx y_i$ *and* $y_i \approx x$*,* $\mathcal{A}$ *is an ABox, and* iexp *is an NI-free expansion strategy, the precondition of the NI-rule is never satisfied.*

5 Two Expansion Strategies

5.1 The $\mathcal{ELOH}$ Case

In this section we present an expansion strategy for the DL $\mathcal{ELOH}$—a logic obtained from $\mathcal{EL}$++ [1] by disallowing role composition axioms $R \circ S \sqsubseteq T$ and concrete domains. We leave these constructors out because they are not available in $\mathcal{SHOIQ}$; however, our results can be straightforwardly extended to full $\mathcal{EL}$++. The expansion strategy for $\mathcal{ELOH}$ will provide us with an intuition on how to approach the general case. Furthermore, these results establish a relationship between model construction algorithms and the implication sets reasoning algorithm from [1]. Thus, tableau DL reasoners can be easily modified to obtain a worst-case optimal decision procedure for $\mathcal{ELOH}$ knowledge bases simply by choosing a suitable individual reuse strategy.

We start by defining $\mathcal{ELOH}$-clauses—the fragment of HT-clauses on which the $\mathcal{ELOH}$-strategy guarantees soundness. By inspecting the translation operator Ξ from [11, Section 4.1], it is easy to see that, if $\mathcal{K}$ is an $\mathcal{ELOH}$ knowledge base, then $\Xi_{\mathcal{TR}}(\mathcal{K})$ is a set of $\mathcal{ELOH}$-clauses.

Definition 5 ($\mathcal{ELOH}$-Clause). *An $\mathcal{ELOH}$-clause is an HT-clause r of the form* (2)*,* (3)*,* (4)*, or* (5) *such that $y_i \neq y_j$ for $i \neq j$; R_j and S are atomic roles; C is of the form $\bot$, A, or $\exists S.A$; A, A_i, and B_j are either $\top$ or atomic but not nominal guard concepts; and O_a is a nominal guard concept.*

$$\bigwedge A_i(x) \wedge \bigwedge [R_j(x, y_j) \wedge B_j(y_j)] \rightarrow C(x) \tag{2}$$

$$O_a(y_a) \wedge \bigwedge A_i(x) \wedge \bigwedge [R_j(x, y_j) \wedge B_j(y_j)] \rightarrow x \approx y_a \tag{3}$$

$$\bigwedge A_i(x) \wedge \bigwedge [R_j(x, y_j) \wedge B_j(y_j)] \rightarrow S(x, y_j) \tag{4}$$

$$O_a(y_a) \wedge \bigwedge A_i(x) \wedge \bigwedge [R_j(x, y_j) \wedge B_j(y_j)] \rightarrow S(x, y_a) \tag{5}$$

We now define the expansion strategy for $\mathcal{ELOH}$.

Definition 6 ($\mathcal{ELHO}$-Strategy). *We assume that, for A an atomic concept, $\top$, or $\bot$, the set of named individuals N_I contains a distinct* representative individual α_A. *The $\mathcal{ELHO}$-strategy is defined as* $\mathsf{iexp}_{EL}(\exists R.A, s, \mathcal{A}) = [(\|\alpha_A\|_{\mathcal{A}})]$, *for each concept $\exists R.A$, individual s, and ABox $\mathcal{A}$.*

Since iexp_{EL} deterministically reuses $\|\alpha_A\|_{\mathcal{A}}$, derivations created using this strategy are typically not safe, so Theorem 1 does not apply. Nonetheless, the strategy always produces sound derivations on $\mathcal{ELHO}$-clauses.

Theorem 2. *For $\mathcal{C}$ a set of $\mathcal{ELHO}$-clauses and $\mathcal{A}$ an initial ABox not containing representative individuals, if $(\mathcal{C}, \mathcal{A})$ is satisfiable, then each derivation for $\mathcal{C}$, $\mathcal{A}$, and iexp_{EL} is successful.*

This theorem can be intuitively understood as follows. Let $\mathcal{C}$ be a set of DL-clauses of the form (2) (we do not consider other types for simplicity) and $\mathcal{A}$ an ABox such that $(\mathcal{C}, \mathcal{A})$ is satisfiable. Assume now that we apply the standard hypertableau calculus without individual reuse and *without blocking* to $\mathcal{C}$ and $\mathcal{A}$. Since blocking is not used, and because all DL-clauses from $\mathcal{C}$ are Horn clauses, this will produce a possibly infinite *canonical* ABox $\mathcal{A}_\infty$ which naturally defines a model of $(\mathcal{C}, \mathcal{A})$. Consider now how individuals are introduced into $\mathcal{A}_\infty$. In particular, assume that an assertion $\exists R.A(s)$ is expanded into assertions $R(s,t)$ and $A(t)$ for t a fresh successor of s. The key observation is that, if the *Hyp*-rule were to additionally derive $C(t)$, then this assertion depends only on the successors of t: a DL-clause of the form (2) can only check properties of successors of the individual mapped to x and not of its predecessors. In other words, the $\mathcal{ELHO}$-clauses do not allow for DL-clauses such as $R(y,x) \rightarrow C(x)$, which might derive C for an individual x based on the properties of its predecessor y. Thus, if in some other part of the model construction we expand $\exists S.A(u)$ into $S(u,v)$ and $A(v)$, the assertions derived for v will depend only on the successors of v. The application of the derivation rules to t and v and their descendants is thus fully determined by the initial "seed" concept A, so t and v will occur in $\mathcal{A}_\infty$ in exactly the same concept assertions (i.e., $C(t) \in \mathcal{A}_\infty$ iff $C(v) \in \mathcal{A}_\infty$). There is, therefore, no need to create distinct individuals t and v at all: we can fold $\mathcal{A}_\infty$ into a model that contains exactly one individual α_A for each "seed" concept A. Effectively, if $(\mathcal{C}, \mathcal{A})$ is satisfiable, then it has such a folded model. It is easy to see that iexp_{EL} essentially constructs exactly this folded model.

The number of representative individuals is linear in the number of the atomic concepts, so the ABoxes constructed in a derivation are polynomial in size. To obtain a polynomial decision procedure, we must be careful in how we apply the *Hyp*-rule to $\mathcal{ELHO}$-clauses: with i individuals and v variables in a DL-clause, a naïve application of the *Hyp*-rule (for each x, for each y_1, for each y_2, and so on) examines i^v combinations. Note, however, that y_i occur in the antecedent only in a tree-like way and that they do not occur in the consequent; hence, we can match each y_i independently, thus giving rise to $v \cdot i$ combinations.

Theorem 3. *A derivation for a set of $\mathcal{ELHO}$-clauses $\mathcal{C}$, an initial ABox $\mathcal{A}$ not containing representative individuals, and iexp_{EL} can be constructed in time polynomial in $|\mathcal{C}, \mathcal{A}|$.*

5.2 The General Case

In this section we define an expansion strategy that is applicable to $\mathcal{SHOIQ}$. The strategy tries to mimic the $\mathcal{ELOH}$ case: it expands $\exists R.A(s)$ into $R(s, \alpha_A)$ and $A(\alpha_A)$ for α_A the representative individual, but it also allows for the default expansion ($*$) as well. Our idea is that, whenever u and v are introduced by expanding concepts $\exists R.A$ and $\exists S.A$, respectively, the individuals u and v should not differ greatly—that is, the derivation rules applied to u and v should be largely the same. For example, it is reasonable to expect that all individuals representing a heart in a model of GALEN should have the same properties. This basic idea has been refined in several ways.

Our experiments have shown that assertions of the form $\exists R.A(s)$, where A is a concept introduced by the structural transformation in the preprocessing phase, are best expanded without individual reuse. Such concepts correspond to complex formulae identifying sets of objects with certain properties, so they are unlikely to have singleton extensions.

Our experiments have also shown that, given $C \sqsubseteq \exists R.D$ and $D \sqsubseteq \exists S.C$, the instance of C implied by the second axiom is usually the instance of C from the first axiom. For example, in GALEN "each artery has a tunica intima as its part," and "each tunica intima is a layer of an artery"; clearly, the second artery is the same as the first one. Therefore, when expanding $\exists S.C(s)$, our strategy tries to reuse the parent t of s if $C(t)$ has already been derived.

DL knowledge bases often contain concepts that should be expanded without reuse—that is, whose expansion using representative individuals usually fails. This can introduce unnecessary nondeterminism, so our strategy learns from failed choices: if an expansion of $\exists R.A$ by reusing α_A fails, our strategy does not reuse α_A in future expansions. The main problem is, once a clash is detected, to determine whether individual reuse caused the inconsistency and, if so, which individual α_A is the culprit. To solve this problem, we construct derivations by depth-first search with dependency-directed backtracking [2, Chapter 9]—a search technique used in most tableau-based reasoners. Roughly speaking, each nondeterministic choice in the derivation is numbered, and each assertion is annotated with a set of choices it depends on. When $\bot$ is derived, the set of choices $S_\bot$ associated with $\bot$ tells us which choices are conflicting. Let $m_\bot$ be the maximal choice from $S_\bot$. Backtracking any choice below $m_\bot$ will invariably lead to a clash; thus, $m_\bot$ identifies the failed choice at which we can safely continue the search. If $m_\bot$ corresponds to a nondeterministic expansion of $\exists R.A$ by reusing α_A, then we consider the reuse of α_A as failed; therefore, we add A to a special *no-good* set of concepts, which ensures that α_A is not reused in future expansions of concepts of the form $\exists S.A$.

Definition 7. *We assume that derivations are constructed by depth-first search and dependency-directed backtracking [2, Chapter 9], and that a set NG of* no-good concepts *is maintained in the process as discussed next. For R a role, B a possibly negated atomic concept, s an individual, and $\mathcal{A}$ an ABox, the* general strategy iexp_{DL} *returns the first result from the following list for which the condition is satisfied.*

- $\mathsf{iexp}_{DL}(\exists R.B, s, \mathcal{A}) = [(t), (*)]$ *if* t *is the parent of* s *and* $B(t) \in \mathcal{A}$.
- $\mathsf{iexp}_{DL}(\exists R.B, s, \mathcal{A}) = [(*)]$ *if* B *has been introduced by the structural transformation during preprocessing.*
- $\mathsf{iexp}_{DL}(\exists R.B, s, \mathcal{A}) = [(\alpha_B), (*)]$ *if* α_B *does not occur in* $\mathcal{A}$ *and* B *is not contained in the current set* NG*. Whenever an expansion using* α_B *is backtracked as per dependency-directed backtracking, the concept* B *is added to the no-good set* NG*.*
- $\mathsf{iexp}_{DL}(\exists R.B, s, \mathcal{A}) = [(\alpha_B)]$ *if* α_B *occurs in* $\mathcal{A}$.
- $\mathsf{iexp}_{DL}(\geq n\, R.B, s, \mathcal{A}) = [(*, \ldots, *)]$ *in all other cases.*

Clearly, iexp_{DL} is bounded. Furthermore, the first time $\exists R.B$ is expanded, iexp_{DL} nondeterministically introduces α_B as a target for future reuse; in all subsequent expansions of $\exists S.B$, iexp_{DL} stays committed to this choice and reuses α_B deterministically. These observations allow us to prove the following proposition.

Proposition 2. *For* $\mathcal{C}$ *a set of HT-clauses and* $\mathcal{A}$ *an initial ABox not containing representative individuals, each derivation for* $\mathcal{C}$*,* $\mathcal{A}$*, and* iexp_{DL} *is safe.*

6 Evaluation

We have implemented individual reuse in HermiT, and have compared its performance with FaCT++ v1.1.10 [16] and Pellet 1.5.1 [12]. When parameterized with iexp_{EL}, the hypertableau calculus becomes quite similar to the implication sets algorithm [1], so any difference in the performance between HermiT and the $\mathcal{EL}$++-specific reasoner CEL is likely to be caused by engineering, rather than algorithmic issues. Therefore, we focus here on evaluating iexp_{DL}.

We have selected several test ontologies commonly used in practice, and have tried to classify them (i.e., to compute $\mathcal{K} \models C \sqsubseteq D$ for all atomic concepts C and D) using HermiT with and without individual reuse, FaCT++, and Pellet. The results are summarized in Table 3. Most unsuccessful tests failed because the reasoners ran out of memory. Several tests were interrupted after 30 minutes; in all such cases, the reasoner got stuck in the first nontrivial subsumption test, so it is unlikely that more time would allow the reasoner to complete the classification. Tests were conducted on a standard laptop PC with 1 GB of RAM running Windows XP. We used Java 1.6.0_04 for Java-based reasoners allowing 600 MB of heap space in each test. All ontologies are available from HermiT's Web page.

Wine. The Wine ontology has often been used to demonstrate the features of description logics. Classifying it was initially a challenge for tableau reasoners, but current reasoners can routinely process it.

FMA-fragment and BAMS. FMA [14] is a large ontology that was not originally developed using a DL, and various translations of different fragments of FMA into DLs have been produced. We used a translation from the Bio-Health Informatics Group at the University of Manchester; it contains 6,487 atomic concepts, 165 atomic roles, 98 individuals, and 18,678 axioms in total. The Brain Architecture Management System (BAMS) is a brain anatomy ontology

Table 3: Summary of Test Results

Ontology	HermiT (with reuse)	HermiT (no reuse)	FaCT++	Pellet
Wine	classified in 37 s	classified in 36 s	classified in 316 s	classified in 25 s
FMA-fragment	classified in 680 s	—	—	—
BAMS	classified in 19 s	—	—	—
DOLCE	classified in 103 s	—	classified in 12 s	—
GALEN-original	classified in 275 s	classified in 26 s	—	—
GALEN-module	—	—	—	—
GALEN-module-no-inv	solves individual tests	—	—	—

produced at the University of Southern California. We used the translation of the 1998 version of BAMS into OWL, which contains 1,110 atomic concepts, 13 atomic roles, 999 individuals, and 20,176 axioms in total. Both ontologies are nontrivial, as they use inverse and functional roles, disjunctions, and nominals.

Without individual reuse, no tool can classify either ontology: reasoners end up constructing large models and quickly exhaust the available memory. In contrast, individual reuse makes these ontologies easy: the models constructed by HermiT consist of a couple of thousands of individuals at most.

DOLCE. DOLCE[3] is a foundational ontology developed in the WonderWeb project, and it is quite complex: it uses disjunctions, nominals, and number restrictions. It is, however, relatively small, containing 211 atomic concepts, 317 atomic roles, 39 individuals, and 1,797 axioms in total. As Table 3 shows, FaCT++ can process DOLCE even without individual reuse, whereas Pellet and HermiT cannot. We conjecture that this is due to ordering optimizations [15], which can have a significant impact on the performance of reasoning. HermiT can process DOLCE provided that individual reuse is turned on. This suggests that individual reuse might compensate for the lack of an optimal ordering, which can be difficult to find in practice.

GALEN. GALEN-original is a version of GALEN dating from about 10 years ago. HermiT is currently the only reasoner that can classify this ontology, and it can do so even without individual reuse [10]. The good performance of HermiT in this case is mainly due to the fact that, after computing a model, HermiT caches the labels of all unblocked individuals and uses them subsequently as potential blockers [11]. Thus, only the first test is hard (it takes about 90% of the total time), and all subsequent tests are easy due to caching. With individual reuse, however, this optimization is ineffective, as most individuals in the model are root individuals that cannot be used as blockers. Each subsumption test takes typically an order of magnitude less time with individual reuse than the "hard" test without reuse; however, the cumulative slowdown due to the lack of caching is detrimental. Caching should be possible, however, with individual reuse as well: we can cache the model constructed in different runs and reuse individuals from it as needed. We shall investigate such a technique in our future work.

[3] `http://www.loa-cnr.it/DOLCE.html`

GALEN was significantly extended in the past 10 years, and the current version of it consists of more than 38,000 concepts, so we have extracted a module from it using the algorithm from [7]. The module is still quite large: it contains 6,362 atomic concepts, 162 atomic roles, and 14,783 axioms in total. None of the reasoners were able to deal with the GALEN-module. We therefore tried removing all axioms of the form $R \sqsubseteq S^-$, effectively eliminating the link between a role and its inverse. After this transformation, HermiT was able to solve individual subsumption tests; however, each test took about 40 s, which made classification of the entire ontology infeasible. We believe that this can be significantly improved using the above mentioned caching technique.

Individual reuse is ineffective on GALEN mainly because the ontology contains numerous functional and inverse-functional roles, which makes identifying concepts suitable for individual reuse difficult. For example, the axiom "each nerve has exactly one axon and vice versa" clearly prevents the reuse of the axon concept. We tried to address this problem by selecting manually the concepts that should be reused, but even this was to no avail. In fact, it seems to us that inverse properties in GALEN are sometimes overconstrained (e.g., w.r.t. functionality). The tools used to develop GALEN largely ignore the semantics of inverse roles, so modeling errors of this kind might easily have gone unnoticed.

7 Conclusion

We have presented a novel calculus for DL reasoning based on individual reuse: instead of always introducing a fresh individual in order to satisfy an existential restriction, our calculus tries to reuse an individual from a model created thus far. Furthermore, we have shown that individual reuse can be done deterministically for $\mathcal{ELOH}$—an expressive but yet tractable DL. Finally, we have implemented our calculus in the reasoner HermiT and have evaluated its performances. The empirical results suggest that individual reuse can mean the difference between success and failure on practical ontologies. The main challenge for our future research is to devise an expansion strategy that will allow us to classify GALEN—the nemesis of DL reasoners.

References

1. F. Baader, S. Brandt, and C. Lutz. Pushing the $\mathcal{EL}$ Envelope. In L. Pack Kaelbling and A. Saffiotti, editors, *Proc. of the 19th Int. Joint Conference on Artificial Intelligence (IJCAI 2005)*, pages 364–369, Edinburgh, UK, July 30–August 5 2005. Morgan Kaufmann Publishers.
2. F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation and Applications*. Cambridge University Press, 2nd edition, August 2007.
3. F. Baader, C. Lutz, and B. Suntisrivaraporn. CEL—A Polynomial-Time Reasoner for Life Science Ontologies. In U. Furbach and N. Shankar, editors, *Proc. of the 3rd Int. Joint Conf. on Automated Reasoning (IJCAR 2006)*, volume 4130 of *LNCS*, pages 287–291, Seattle, WA, USA, August 17–20 2006. Springer.

4. P. Baumgartner, U. Furbach, and I. Niemelä. Hyper Tableaux. In *Proc. of the European Workshop on Logics in Artificial Intelligence (JELIA '96)*, number 1126 in LNAI, pages 1–17, Évora, Portugal, September 30–October 3 1996. Springer.
5. F. Bry and S. Torge. A Deduction Method Complete for Refutation and Finite Satisfiability. In J. Dix, L. F. del Cerro, and U. Furbach, editors, *Proc. European Workshop on Logics in Artificial Intelligence (JELIA '98)*, volume 1489 of *LNCS*, pages 122–138, Dagstuhl, Germany, October 12–15 1998. Springer.
6. L. Georgieva, U. Hustadt R. A., and Schmidt. Hyperresolution for Guarded Formulae. *Journal of Symbolic Computation*, 36(1–2):163–192, 2003.
7. B. Cuenca Grau, I. Horrocks, Y. Kazakov, and U. Sattler. Just the Right Amount: Extracting Modules from Ontologies. In *Proc. of the 16th Int. Conf. on World Wide Web (WWW 2007)*, pages 717–726, Banff, AB, Canada, May 8–12 2007. ACM Press.
8. V. Haarslev and R. Möller. RACER System Description. In R. Goré, A. Leitsch, and T. Nipkow, editors, *Proc. of the 1st Int. Joint Conf. on Automated Reasoning (IJCAR 2001)*, volume 2083 of *LNAI*, pages 701–706, Siena, Italy, June 18–23 2001. Springer.
9. U. Hustadt and R. A. Schmidt. Issues of Decidability for Description Logics in the Framework of Resolution. In R. Caferra and G. Salzer, editors, *Selected Papers from Automated Deduction in Classical and Non-Classical Logics*, volume 1761 of *LNAI*, pages 191–205. Springer, 1999.
10. B. Motik, R. Shearer, and I. Horrocks. Optimized Reasoning in Description Logics using Hypertableaux. In F. Pfenning, editor, *Proc. of the 21st Conference on Automated Deduction (CADE-21)*, volume 4603 of *LNAI*, pages 67–83, Bremen, Germany, July 17–20 2007. Springer.
11. B. Motik, R. Shearer, and I. Horrocks. Hypertableau Reasoning for Description Logics. Technical report, University of Oxford, 2008. Submitted to an international journal.
12. B. Parsia and E. Sirin. Pellet: An OWL-DL Reasoner. Poster, In Proc. of the 3rd Int. Semantic Web Conference (ISWC 2004), Hiroshima, Japan, November 7–11, 2004.
13. A. L. Rector, W. A. Nowlan, and A. Glowinski. Goals for concept representation in the galen project. In C. Safran, editor, *Proc. of the 17th Annual Symposium on Computer Applications in Medical Care (SCAMC '93)*, pages 414–418, Washington DC, USA, November 1–3 1993. McGraw-Hill.
14. C. Rosse and J. V. L. Mejino. A reference ontology for biomedical informatics: the Foundational Model of Anatomy. *Journal of Biomedical Informatics*, 36:478–500, 2003.
15. D. Tsarkov and I. Horrocks. Ordering Heuristics for Description Logic Reasoning. In L. Pack Kaelbling and A. Saffiotti, editors, *Proc. of the 19th Int. Joint Conf. on Artificial Intelligence (IJCAI 2005)*, pages 609–614, Edinburgh, UK, July 30–August 5 2005. Morgan Kaufmann Publishers.
16. D. Tsarkov and I. Horrocks. FaCT++ Description Logic Reasoner: System Description. In *Proc. of the 3rd Int. Joint Conf. on Automated Reasoning (IJCAR 2006)*, volume 4130 of *LNAI*, pages 292–297, Seattle, WA, USA, August 17–20 2006. Springer.
17. M. Y. Vardi. Why Is Modal Logic So Robustly Decidable? In N. Immerman and P. Kolaitis, editors, *Proc. of a DIMACS Workshop on Descriptive Complexity and Finite Models*, volume 31 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pages 149–184, Princeton University, USA, January 14–17 1996. American Mathematical Society.

A Proofs of Theorems

Proof (of Theorem 1). Due to Definition 1, the $\geq$-rule introduces only assertions of the form $R(s,t)$ where t or s is a root individual, or one individual is the predecessor of the other. Therefore, each ABox labeling a derivation node is tree-shaped, c.f. [11, Definition 11] and [11, Lemma 12].

(Claim 1) Since iexp is bounded, the number of fresh individuals introduced by the $\geq$-rule due to individual reuse is finite in each derivation. The number of root and blockable individuals can then be bounded as in [11, Lemma 15].

(Claim 2) If $\mathcal{A}'$ is a clash-free ABox labeling a leaf of a successful derivation, it is tree-shaped, so a model of $\mathcal{A}'$ and $\mathcal{C}$ can be extracted as in [11, Lemma 14].

(Claim 3) Let $D = (T, \lambda)$ be a safe derivation for $\mathcal{C}$, $\mathcal{A}$, and iexp, and consider a derivation D' obtained from D by deleting all unsafe nodes starting from the root. The conditions of Definition 3 guarantee that each node $t \in T$ obtained by the $\geq$-rule has at least one safe child. Therefore, D' is similar to a derivation obtained without individual reuse, with the minor difference that fresh named individuals can be used to satisfy existential restrictions. We can always interpret these individuals as required, so, just like in [11, Lemma 13], if $(\mathcal{C}, \mathcal{A})$ is satisfiable, then D' is successful; but then, D is successful as well. □

Proof (of Proposition 1). Since iexp is *NI*-free, the $\geq$-rule does not introduce assertions of the form $R(s,a)$ or $R(a,s)$ for a a root individual and s a blockable individual that is not a successor of a. As shown in [11, Lemma 12], no other rule can then introduce them either. But then, an at-most equality satisfying the precondition of the *NI*-rule cannot be derived either. □

Proof (of Theorem 2). Let $\mathcal{A} = \mathcal{A}_0, \mathcal{A}_1, \ldots, \mathcal{A}_n, \ldots$ be a derivation for $\mathcal{C}$, $\mathcal{A}$, and iexp_{EL}, and $[\cdot]$ a function on N_I such that $[\alpha_A] = A$ and $[s] = \{s\}$ if s is not a representative individual. We show inductively that, if $(\mathcal{C}, \mathcal{A})$ is satisfiable, then the following properties (*) hold for each ABox $\mathcal{A}_n$ in the derivation and each model I of $(\mathcal{C}, \mathcal{A})$:

(i)	$\|s\|_{\mathcal{A}_n} = \|t\|_{\mathcal{A}_n} \Rightarrow [t]^I = [s]^I$	(v)	$C(s) \in \mathcal{A}_n \Rightarrow [s]^I \subseteq C^I$
(ii)	$s \approx t \in \mathcal{A}_n \Rightarrow [s]^I = [t]^I$	(vi)	$R(s,t) \in \mathcal{A}_n \Rightarrow [s]^I \subseteq (\exists R.[t])^I$
(iii)	$s \not\approx t \in \mathcal{A}_n \Rightarrow [s]^I \cap [t]^I = \emptyset$	(vii)	$\bot \notin \mathcal{A}_n$
(iv)	s *occurs in* $\mathcal{A}_n \Rightarrow [s]^I \neq \emptyset$		

The ABox $\mathcal{A}_0$ trivially satisfies (*). Assume that $\mathcal{A}_n$ satisfies (*) and, for each derivation rule applicable to $\mathcal{A}_n$, consider a model I of $(\mathcal{C}, \mathcal{A}_n)$. The *NI*-rule is never applicable since $\mathcal{C}$ does not contain at-most equalities. If the $\bot$-rule were applicable to $A(s)$ and $\neg A(s)$ (resp. $s \not\approx s$), by (*) we have $[s]^I \neq \emptyset$, $[s]^I \subseteq A^I$, and $[s]^I \subseteq (\neg A)^I$ (resp. $[s]^I \neq \emptyset$ and $[s]^I \cap [s]^I = \emptyset$), which is a contradiction.

Assume that the $\approx$-rule is applied to $s \approx t \in \mathcal{A}_n$ and s is merged into t. By (*) we have $[s]^I = [t]^I$. Clearly, $\|s\|_{\mathcal{A}_n} = t$, and each assertion obtained by replacing s with t in an assertion from $\mathcal{A}_n$ clearly satisfies (*).

Assume that the $\exists$-rule is applied to $\exists R.A(s) \in \mathcal{A}_n$, so the assertions $R(s,t)$ and $A(t)$ are added to $\mathcal{A}_{n+1}$ for $t = \|\alpha_A\|_{\mathcal{A}_n}$. By (*) we have $[s]^I \subseteq (\exists R.A)^I$ and $[t]^I = [\alpha_A]^I = A^I \neq \emptyset$; this obviously implies $[s]^I \subseteq (\exists R.[t])^I$ and $[t]^I \subseteq A^I$.

Assume that the *Hyp*-rule is applied to a DL-clause r of the form (2)–(5), and let $s = \sigma(x)$, $t_n = \sigma(y_n)$, and $t_a = \sigma(y_a)$. By Condition 2.2 of the *Hyp*-rule, we have $A_i(s) \in \mathcal{A}_n$, $R_j(s, t_j) \in \mathcal{A}_n$, $B_j(t_j) \in \mathcal{A}_n$, and $O_a(t_a) \in \mathcal{A}_n$ (if O_a is present in r), so all these assertions satisfy (*). Consider now each $\delta \in [s]^I$, and let μ be the following mapping from the variables of r to the elements of $\triangle^I$: $\mu(x) = \delta$, $\mu(y_j)$ are arbitrarily chosen such that $\langle \mu(x), \mu(y_j) \rangle \in R^I$ (such objects exist by (vi) and the fact that $y_i \neq y_j$ whenever $i \neq j$), and $\mu(y_a) \in [t_a]^I$ (such object exists by (iv)). By (*) then $\mu(x) \in A_i^I$ and $\mu(y_j) \in B_j^I$. Since I is a model of r, the head atom of r must then be true in I, so it is not $\bot$. We now consider different forms that r can have. For r of the form (2), $I \models C(\mu(x))$ implies $\delta \in C^I$; this holds for each $\delta \in [s]^I$, so $[s]^I \subseteq C^I$. For r of the form (3), $I \models \mu(x) \approx \mu(y_a)$ implies $\delta = \mu(y_a)$; this holds for each $\delta \in [s]^I$ and each $\mu(y_a) \in [t_a]^I$, so $[s]^I = [t_a]^I$ For r of the form (4), $I \models S(\mu(x), \mu(y_j))$ implies $\langle \delta, \mu(y_j) \rangle \in S^I$; this holds for each $\delta \in [s]^I$, so $[s]^I \subseteq (\exists S.[t_j])^I$. For r of the form (5), $I \models S(\mu(x), \mu(y_a))$ implies $\langle \delta, \mu(y_a) \rangle \in S^I$; this holds for each $\delta \in [s]^I$ and each $\mu(y_a) \in [t_a]^I$, so $[s]^I \subseteq (\exists S.[t_a])^I$. □

Proof (of Theorem 3). In each a derivation on $\mathcal{C}$ and $\mathcal{A}$, the strategy iexp_{EL} can introduce at most one representative individual for each concept, so the total number of individuals i in each ABox is linear in $|\mathcal{C}, \mathcal{A}|$, giving rise to polynomially many different assertions. A derivation rule is never applied twice to the same set of assertions [11, Lemma 15]. The $\bot$-, $\exists$-, and $\approx$-rule can clearly be applied to an ABox in polynomial time. Let v be the maximal number of variables in an $\mathcal{ELOH}$-clause. To avoid an exponential blowup, we can match a DL-clause r to an ABox as follows: we map x to each of the i individuals; if r contains O_a, we additionally examine each of the possible i mappings for y_a; finally, we can match the variables y_i independent of each other. Using such a strategy, the *Hyp*-rule needs to examine at most $i \cdot i \cdot vi$ substitutions, which is polynomial in $|\mathcal{C}, \mathcal{A}|$. □

Proof (of Proposition 2). A derivation node t can be nonsafe only if it has been obtained by an application of the $\geq$-rule and the expansion $[(\alpha_A)]$ (the fourth case of Definition 7). Since the initial ABox $\mathcal{A}$ does not contain α_A, an ancestor node t' of t exists that is obtained by an application of the $\geq$-rule with the expansion $[(\alpha_A), (*)]$. Node t' is safe and so is its second child, so the derivation is safe as well. □