Exploiting Pseudo Models for TBox and ABox Reasoning in Expressive Description Logics

Volker Haarslev* and Ralf Möller* and Anni-Yasmin Turhan**

* University of Hamburg
Computer Science Department
Vogt-Kölln-Str. 30
22527 Hamburg, Germany

** RWTH Aachen
LuFG Theoretical Computer Science
Ahornstr. 55
52074 Aachen, Germany

Abstract. This paper investigates optimization techniques and data structures exploiting the use of so-called *pseudo models*. These techniques are applied to speed up TBox and ABox reasoning for the description logics $\mathcal{ALCNH}_{R^+}$ and $\mathcal{ALC}(\mathcal{D})$. The advances are demonstrated by an empirical analysis using the description logic system RACE that implements TBox and ABox reasoning for $\mathcal{ALCNH}_{R^+}$.

1 Introduction

We introduce and analyze optimization techniques for reasoning in expressive description logics exploiting so-called pseudo models. The new techniques being investigated are called *deep model merging* and *individual model merging*. The presented algorithms are empirically evaluated using TBoxes and ABoxes derived from actual applications. The model merging technique is also developed for the logic $\mathcal{ALC}(\mathcal{D})$ [1] which supports so-called concrete domains. This is motivated by a proposal which extends $\mathcal{ALCNH}_{R^+}$ with a restricted form of concrete domains [4].

1.1 The Language $\mathcal{ALCNH}_{R^+}$

We briefly introduce the description logic (DL) $\mathcal{ALCNH}_{R^+}$ [3] (see the tables in Figure 1) using a standard Tarski-style semantics based on an interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ $\mathcal{ALCNH}_{R^+}$ extends the basic description logic $\mathcal{ALC}$ by role hierarchies, transitively closed roles, and number restrictions. Note that the combination of transitive roles and role hierarchies implies the expressiveness of so-called general inclusion axioms (GCIs). The language definition is slightly extended compared to the one given in [3] since we additionally support the declaration of "native" features. This allows additional optimizations, e.g. an efficient treatment of features by the model merging technique (see below). The concept name $\top$ ($\bot$) is used as an abbreviation for $\mathsf{C} \sqcup \neg\mathsf{C}$ ($\mathsf{C} \sqcap \neg\mathsf{C}$). We assume a set of concept names C, a set of role names R, and a set of individual names O. The mutually disjoint subsets F, P, T of R denote features, non-transitive, and transitive roles, respectively ($R = F \cup P \cup T$).

(a)

Syntax	Semantics
Concepts	
A	$\mathsf{A}^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$
$\neg\mathsf{C}$	$\Delta^{\mathcal{I}} \setminus \mathsf{C}^{\mathcal{I}}$
$\mathsf{C} \sqcap \mathsf{D}$	$\mathsf{C}^{\mathcal{I}} \cap \mathsf{D}^{\mathcal{I}}$
$\mathsf{C} \sqcup \mathsf{D}$	$\mathsf{C}^{\mathcal{I}} \cup \mathsf{D}^{\mathcal{I}}$
$\exists\,\mathsf{R}\,.\,\mathsf{C}$	$\{\mathsf{a} \in \Delta^{\mathcal{I}} \mid \exists\,\mathsf{b} \in \Delta^{\mathcal{I}} : (\mathsf{a},\mathsf{b}) \in \mathsf{R}^{\mathcal{I}},\ \mathsf{b} \in \mathsf{C}^{\mathcal{I}}\}$
$\forall\,\mathsf{R}\,.\,\mathsf{C}$	$\{\mathsf{a} \in \Delta^{\mathcal{I}} \mid \forall\,\mathsf{b} : (\mathsf{a},\mathsf{b}) \in \mathsf{R}^{\mathcal{I}} \Rightarrow \mathsf{b} \in \mathsf{C}^{\mathcal{I}}\}$
$\exists_{\geq n}\,\mathsf{S}$	$\{\mathsf{a} \in \Delta^{\mathcal{I}} \mid \|\{\mathsf{b} \in \Delta^{\mathcal{I}} \mid (\mathsf{a},\mathsf{b}) \in \mathsf{S}^{\mathcal{I}}\}\| \geq \mathsf{n}\}$
$\exists_{\leq m}\,\mathsf{S}$	$\{\mathsf{a} \in \Delta^{\mathcal{I}} \mid \|\{\mathsf{b} \in \Delta^{\mathcal{I}} \mid (\mathsf{a},\mathsf{b}) \in \mathsf{S}^{\mathcal{I}}\}\| \leq \mathsf{m}\}$
Roles	
R	$\mathsf{R}^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$

(b)

Terminol. Axioms	
Syntax	Satisfied if
$\mathsf{R} \in T$	$\mathsf{R}^{\mathcal{I}} = (\mathsf{R}^{\mathcal{I}})^{+}$
$\mathsf{F} \in F$	$\Delta^{\mathcal{I}} \subseteq (\exists_{\leq 1}\,\mathsf{F})^{\mathcal{I}}$
$\mathsf{R} \sqsubseteq \mathsf{S}$	$\mathsf{R}^{\mathcal{I}} \subseteq \mathsf{S}^{\mathcal{I}}$
$\mathsf{C} \sqsubseteq \mathsf{D}$	$\mathsf{C}^{\mathcal{I}} \subseteq \mathsf{D}^{\mathcal{I}}$

(c)

Assertions	
Syntax	Satisfied if
$\mathsf{a} : \mathsf{C}$	$\mathsf{a}^{\mathcal{I}} \in \mathsf{C}^{\mathcal{I}}$
$(\mathsf{a},\mathsf{b}) : \mathsf{R}$	$(\mathsf{a}^{\mathcal{I}}, \mathsf{b}^{\mathcal{I}}) \in \mathsf{R}^{\mathcal{I}}$

Fig. 1. Syntax and Semantics of $\mathcal{ALCNH}_{R^+}$ ($n, m \in \mathbb{N}$, $n > 0$, $\|\cdot\|$ denotes set cardinality, and $\mathsf{S} \in S$).

If $\mathsf{R}, \mathsf{S} \in R$ are role names, then the terminological axiom $\mathsf{R} \sqsubseteq \mathsf{S}$ is called a *role inclusion axiom*. A *role hierarchy* $\mathcal{R}$ is a finite set of role inclusion axioms. Then, we define $\sqsubseteq^*$ as the reflexive transitive closure of $\sqsubseteq$ over such a role hierarchy $\mathcal{R}$. Given $\sqsubseteq^*$, the set of roles $\mathsf{R}^{\downarrow} = \{\mathsf{S} \in R \mid \mathsf{S} \sqsubseteq^* \mathsf{R}\}$ defines the *descendants* of a role R. $\mathsf{R}^{\uparrow} = \{\mathsf{S} \in R \mid \mathsf{R} \sqsubseteq^* \mathsf{S}\}$ is the set of *ancestors* of a role R. We also define the set $S = \{\mathsf{R} \in P \mid \mathsf{R}^{\downarrow} \cap T = \emptyset\}$ of *simple* roles that are neither transitive nor have a transitive role as descendant. Every descendant G of a feature F must be a feature as well ($\mathsf{G} \in F$).

A syntactic restriction holds for the combinability of number restrictions and transitive roles in $\mathcal{ALCNH}_{R^+}$. Number restrictions are only allowed for *simple* roles. This restriction is motivated by an undecidability result in case of an unrestricted combinability [8].

If C and D are concept terms, then $\mathsf{C} \sqsubseteq \mathsf{D}$ (*generalized concept inclusion* or *GCI*) is a terminological axiom. A finite set of terminological axioms $\mathcal{T}_{\mathcal{R}}$ is called a *terminology* or *TBox* w.r.t. to a given role hierarchy $\mathcal{R}$.[1]

An *ABox* $\mathcal{A}$ is a finite set of assertional axioms as defined in Figure 1c. The set O of object names is divided into two disjoint subsets, O_O and O_N.[2] An initial ABox $\mathcal{A}$ may contain only assertions mentioning old individuals (from O_O). Every individual name from O is mapped to a single element of $\Delta^{\mathcal{I}}$ in a way such that for $\mathsf{a}, \mathsf{b} \in O_O$, $\mathsf{a}^{\mathcal{I}} \neq \mathsf{b}^{\mathcal{I}}$ if $\mathsf{a} \neq \mathsf{b}$ *(unique name assumption* or UNA). This ensures that different individuals in O_O are interpreted as different elements. The UNA does not hold for elements of O_N, i.e. for $\mathsf{a}, \mathsf{b} \in O_N$, $\mathsf{a}^{\mathcal{I}} = \mathsf{b}^{\mathcal{I}}$ may hold even if $\mathsf{a} \neq \mathsf{b}$, or if we assume without loss of generality that $\mathsf{a} \in O_N, \mathsf{b} \in O_O$.

The *ABox consistency problem* is to decide whether a given ABox $\mathcal{A}$ is consistent w.r.t. a TBox $\mathcal{T}$. Satisfiability of concept terms can be reduced to ABox

[1] The reference to $\mathcal{R}$ is omitted in the following.

[2] The set of "old" individuals names characterizes all individuals for which the unique name assumption holds while the set of "new" names denotes individuals which are constructed during a proof.

consistency as follows: A concept term C is *satisfiable* iff the ABox $\{\mathsf{a}:\mathsf{C}\}$ is consistent. The *instance problem* is to determine whether an individual a is an instance of a concept term C w.r.t. an ABox $\mathcal{A}$ and a TBox $\mathcal{T}$, i.e. whether $\mathcal{A}$ entails $\mathsf{a}:\mathsf{C}$ w.r.t. $\mathcal{T}$. This problem can be reduced to the problem of deciding if the ABox $\mathcal{A} \cup \{\mathsf{a}:\neg\mathsf{C}\}$ is inconsistent w.r.t. $\mathcal{T}$.

1.2 A Tableaux Calculus for $\mathcal{ALCNH}_{R^+}$

In the following we present a *tableaux algorithm* to decide the consistency of $\mathcal{ALCNH}_{R^+}$ ABoxes. The algorithm is characterized by a set of tableaux or *completion rules* and by a particular *completion strategy* ensuring a specific order for applying the completion rules to assertional axioms of an ABox. The strategy is essential to guarantee the completeness of the ABox consistency algorithm. The purpose of the calculus is to generate a so-called completion for an initial ABox $\mathcal{A}$ in order to prove the consistency of $\mathcal{A}$ or its inconsistency if no completion can be found.

First, we introduce new assertional axioms needed to define the augmentation of an initial ABox. Let C be a concept term, $\mathsf{a}, \mathsf{b} \in O$ be individual names, and $x \notin O$, then the following expressions are also assertional axioms: (1) $\forall x \,.\, (x:\mathsf{C})$ *(universal concept assertion)*, (2) $\mathsf{a} \not\doteq \mathsf{b}$ *(inequality assertion)*. An interpretation $\mathcal{I}$ satisfies an assertional axiom $\forall x \,.\, (x:\mathsf{C})$ iff $\mathsf{C}^{\mathcal{I}} = \Delta^{\mathcal{I}}$ and $\mathsf{a} \not\doteq \mathsf{b}$ iff $\mathsf{a}^{\mathcal{I}} \neq \mathsf{b}^{\mathcal{I}}$.

We are now ready to define an augmented ABox as input to the tableaux rules. For an initial ABox $\mathcal{A}$ w.r.t a TBox $\mathcal{T}$ and a role hierarchy $\mathcal{R}$ we define its *augmented* ABox or its *augmentation* $\mathcal{A}'$ by applying the following rules to $\mathcal{A}$. For every feature name F mentioned in $\mathcal{A}$ the assertion $\forall x \,.\, (x:(\exists_{\leq 1}\, \mathsf{F}))$ is added to $\mathcal{A}'$. For every GCI $\mathsf{C} \sqsubseteq \mathsf{D}$ in $\mathcal{T}$ the assertion $\forall x \,.\, (x:(\neg\mathsf{C} \sqcup \mathsf{D}))$ is added to $\mathcal{A}'$. Every concept term occurring in $\mathcal{A}$ is transformed into its usual negation normal form. Let $O'_O = \{\mathsf{a}_1, \ldots, \mathsf{a}_\mathsf{n}\} \subseteq O_O$ be the set of individuals mentioned in $\mathcal{A}$, then the following set of inequality assertions is added to $\mathcal{A}'$: $\{\mathsf{a}_\mathsf{i} \not\doteq \mathsf{a}_\mathsf{j} \mid \mathsf{a}_\mathsf{i}, \mathsf{a}_\mathsf{j} \in O'_O,\ \mathsf{i}, \mathsf{j} \in 1..\mathsf{n},\ \mathsf{i} \neq \mathsf{j}\}$. Obviously, if $\mathcal{A}'$ is an augmentation of $\mathcal{A}$ then $\mathcal{A}'$ is consistent iff $\mathcal{A}$ is consistent.

$\mathcal{ALCNH}_{R^+}$ supports transitive roles and GCIs. Thus, in order to guarantee the termination of the tableaux calculus, the notion of *blocking* an individual for the applicability of tableaux rules is introduced as follows. Given an ABox $\mathcal{A}$ and an individual a occurring in $\mathcal{A}$, we define the *concept set* of a as $\sigma(\mathcal{A}, \mathsf{a}) := \{\top\} \cup \{\mathsf{C} \mid \mathsf{a}:\mathsf{C} \in \mathcal{A}\}$. We define an *individual ordering* '$\prec$' for new individuals (elements of O_N) occurring in an ABox $\mathcal{A}$. If $\mathsf{b} \in O_N$ is introduced into $\mathcal{A}$, then $\mathsf{a} \prec \mathsf{b}$ for all new individuals a already present in $\mathcal{A}$. Let $\mathcal{A}$ be an ABox and $\mathsf{a}, \mathsf{b} \in O$ be individuals in $\mathcal{A}$. We call a the blocking individual of b if all of the following conditions hold: (1) $\mathsf{a}, \mathsf{b} \in O_N$, (2) $\sigma(\mathcal{A}, \mathsf{a}) \supseteq \sigma(\mathcal{A}, \mathsf{b})$, (3) $\mathsf{a} \prec \mathsf{b}$. If there exists a blocking individual a for b, then b is said to be *blocked* (by a).

We are now ready to define the *completion rules* that are intended to generate a so-called completion (see also below) of an initial ABox $\mathcal{A}$ w.r.t. a TBox $\mathcal{T}$.

R$\sqcap$ The conjunction rule.
  if $\mathsf{a}:\mathsf{C}\sqcap\mathsf{D}\in\mathcal{A}$, and $\{\mathsf{a}:\mathsf{C},\ \mathsf{a}:\mathsf{D}\}\not\subseteq\mathcal{A}$
  then $\mathcal{A}'=\mathcal{A}\cup\{\mathsf{a}:\mathsf{C},\ \mathsf{a}:\mathsf{D}\}$

R$\sqcup$ The disjunction rule.
  if $\mathsf{a}:\mathsf{C}\sqcup\mathsf{D}\in\mathcal{A}$, and $\{\mathsf{a}:\mathsf{C},\ \mathsf{a}:\mathsf{D}\}\cap\mathcal{A}=\emptyset$
  then $\mathcal{A}'=\mathcal{A}\cup\{\mathsf{a}:\mathsf{C}\}$ **or** $\mathcal{A}'=\mathcal{A}\cup\{\mathsf{a}:\mathsf{D}\}$

R$\forall$C The role value restriction rule.
  if $\mathsf{a}:\forall\,\mathsf{R}\,.\,\mathsf{C}\in\mathcal{A}$, and $\exists\,\mathsf{b}\in O,\mathsf{S}\in\mathsf{R}^{\downarrow}:(\mathsf{a},\mathsf{b}):\mathsf{S}\in\mathcal{A}$, and $\mathsf{b}:\mathsf{C}\notin\mathcal{A}$
  then $\mathcal{A}'=\mathcal{A}\cup\{\mathsf{b}:\mathsf{C}\}$

R$\forall_+$C The transitive role value restriction rule.
  if 1. $\mathsf{a}:\forall\,\mathsf{R}\,.\,\mathsf{C}\in\mathcal{A}$, and $\exists\,\mathsf{b}\in O$, $\mathsf{T}\in\mathsf{R}^{\downarrow}$, $\mathsf{T}\in T$, $\mathsf{S}\in\mathsf{T}^{\downarrow}:(\mathsf{a},\mathsf{b}):\mathsf{S}\in\mathcal{A}$, and
  2. $\mathsf{b}:\forall\,\mathsf{T}\,.\,\mathsf{C}\notin\mathcal{A}$
  then $\mathcal{A}'=\mathcal{A}\cup\{\mathsf{b}:\forall\,\mathsf{T}\,.\,\mathsf{C}\}$

R$\forall_x$ The universal concept restriction rule.
  if $\forall\,x\,.\,(x:\mathsf{C})\in\mathcal{A}$, and $\exists\,\mathsf{a}\in O$: a mentioned in $\mathcal{A}$, and $\mathsf{a}:\mathsf{C}\notin\mathcal{A}$
  then $\mathcal{A}'=\mathcal{A}\cup\{\mathsf{a}:\mathsf{C}\}$

R$\exists$C The role exists restriction rule.
  if 1. $\mathsf{a}:\exists\,\mathsf{R}\,.\,\mathsf{C}\in\mathcal{A}$, and a is not blocked, and
  2. $\neg\exists\,\mathsf{b}\in O$, $\mathsf{S}\in\mathsf{R}^{\downarrow}:\{(\mathsf{a},\mathsf{b}):\mathsf{S},\ \mathsf{b}:\mathsf{C}\}\subseteq\mathcal{A}$
  then $\mathcal{A}'=\mathcal{A}\cup\{(\mathsf{a},\mathsf{b}):\mathsf{R},\ \mathsf{b}:\mathsf{C}\}$ where $\mathsf{b}\in O_N$ is not used in $\mathcal{A}$

R$\exists_{\geq n}$ The number restriction exists rule.
  if 1. $\mathsf{a}:\exists_{\geq n}\,\mathsf{R}\in\mathcal{A}$, and a is not blocked, and
  2. $\neg\exists\,\mathsf{b}_1,\ldots,\mathsf{b}_\mathsf{n}\in O$, $\mathsf{S}_1,\ldots,\mathsf{S}_\mathsf{n}\in\mathsf{R}^{\downarrow}$:
  $\{(\mathsf{a},\mathsf{b}_\mathsf{k}):\mathsf{S}_\mathsf{k}\mid\mathsf{k}\in 1..\mathsf{n}\}\cup\{\mathsf{b}_\mathsf{i}\not\doteq\mathsf{b}_\mathsf{j}\mid\mathsf{i},\mathsf{j}\in 1..\mathsf{n},\mathsf{i}\neq\mathsf{j}\}\subseteq\mathcal{A}$
  then $\mathcal{A}'=\mathcal{A}\cup\{(\mathsf{a},\mathsf{b}_\mathsf{k}):\mathsf{R}\mid\mathsf{k}\in 1..\mathsf{n}\}\cup\{\mathsf{b}_\mathsf{i}\not\doteq\mathsf{b}_\mathsf{j}\mid\mathsf{i},\mathsf{j}\in 1..\mathsf{n},\mathsf{i}\neq\mathsf{j}\}$
  where $\mathsf{b}_1,\ldots,\mathsf{b}_\mathsf{n}\in O_N$ are not used in $\mathcal{A}$

R$\exists_{\leq n}$ The number restriction merge rule.
  if 1. $\mathsf{a}:\exists_{\leq n}\,\mathsf{R}\in\mathcal{A}$, and
  2. $\exists\,\mathsf{b}_1,\ldots,\mathsf{b}_\mathsf{m}\in O$, $\mathsf{S}_1,\ldots,\mathsf{S}_\mathsf{m}\in\mathsf{R}^{\downarrow}$: $\{(\mathsf{a},\mathsf{b}_1):\mathsf{S}_1,\ldots,(\mathsf{a},\mathsf{b}_\mathsf{m}):\mathsf{S}_\mathsf{m}\}\subseteq\mathcal{A}$
  with $\mathsf{m}>\mathsf{n}$, and
  3. $\exists\,\mathsf{b}_\mathsf{i},\mathsf{b}_\mathsf{j}\in\{\mathsf{b}_1,\ldots,\mathsf{b}_\mathsf{m}\}:\mathsf{i}\neq\mathsf{j},\ \mathsf{b}_\mathsf{i}\not\doteq\mathsf{b}_\mathsf{j}\notin\mathcal{A}$
  then $\mathcal{A}'=\mathcal{A}[\mathsf{b}_\mathsf{i}/\mathsf{b}_\mathsf{j}]$, i.e. replace every occurrence of b_i in $\mathcal{A}$ by b_j

Given an ABox $\mathcal{A}$, more than one rule might be applicable to $\mathcal{A}$. The order is determined by the *completion strategy* which is defined as follows.

A *meta rule* controls the priority between individuals: Apply a tableaux rule to an individual $\mathsf{b}\in O_N$ only if no rule is applicable to an individual $\mathsf{a}\in O_O$ and if no rule is applicable to another individual $\mathsf{c}\in O_N$ such that $\mathsf{c}\prec\mathsf{b}$.

The completion rules are always applied in the following order. (1) Apply all non-generating rules (R$\sqcap$, R$\sqcup$, R$\forall$C, R$\forall_+$C, R$\forall_x$, R$\exists_{\leq n}$) as long as possible. (2) Apply a generating rule (R$\exists$C, R$\exists_{\geq n}$) once and continue with step 1.

In the following we always assume that the completion strategy is observed. This ensures that rules are applied to new individuals w.r.t. the ordering '$\prec$'.

We assume the same naming conventions as used above. An ABox $\mathcal{A}$ is called *contradictory* if one of the following *clash triggers* is applicable. If none of the clash triggers is applicable to $\mathcal{A}$, then $\mathcal{A}$ is called *clash-free*. The clash triggers have to deal with so-called primitive clashes and with clashes caused by number restrictions:

- *Primitive clash*: $\mathsf{a}:\bot \in \mathcal{A}$ or $\{\mathsf{a}:\mathsf{A}, \mathsf{a}:\neg\mathsf{A}\} \subseteq \mathcal{A}$, where A is a concept name.
- *Number restriction merging clash*: $\exists\, \mathsf{S}_1, \ldots, \mathsf{S}_\mathsf{m} \in \mathsf{R}^{\downarrow} : \{(\mathsf{a}, \mathsf{b}_\mathsf{i}):\mathsf{S}_\mathsf{i} \mid \mathsf{i} \in 1..\mathsf{m}\} \cup \{\mathsf{a}:\exists_{\leq n}\, \mathsf{R}\} \cup \{\mathsf{b}_\mathsf{i} \not\doteq \mathsf{b}_\mathsf{j} \mid \mathsf{i}, \mathsf{j} \in 1..\mathsf{m}, \mathsf{i} \neq \mathsf{j}\} \subseteq \mathcal{A}$ with $\mathsf{m} > \mathsf{n}$.

Any ABox containing a clash is obviously unsatisfiable. A clash-free ABox $\mathcal{A}$ is called *complete* if no completion rule is applicable to $\mathcal{A}$. A complete ABox $\mathcal{A}'$ derived from an initial ABox $\mathcal{A}$ is called a *completion* of $\mathcal{A}$. The purpose of the calculus is to generate a completion for an initial ABox $\mathcal{A}$ to prove the consistency of $\mathcal{A}$. An augmented ABox $\mathcal{A}$ is said to be inconsistent if no completion can be derived. For a given initial ABox $\mathcal{A}$, the calculus applies the completion rules. It stops the application of rules, if a clash occurs. The calculus answers "*yes*" if a completion can be derived, and "*no*" otherwise. Based on these notions we introduce and evaluate the new optimization techniques in the next sections.

2 Deep Models for TBox Reasoning in $\mathcal{ALCNH}_{R^+}$

Given a set of concepts representing a conjunction whose satisfiability is to be checked, the *model merging strategy* tries to avoid a satisfiability test which relies on the "expensive" tableaux technique due to non-deterministic rules.[3] This idea was first introduced in [5] for the logic $\mathcal{ALCH}f_{R^+}$. A model merging test is designed to be a "cheap" test comparing cached "concept models." It is a *sound* but incomplete satisfiability tester for a set of concepts. The achievement of minimal computational overhead and the avoidance of any indeterminism are important characteristics of such a test. If the test returns false, a tableaux calculus based on the rules as defined in Section 1.2 is applied. In order to be more precise, we use the term *pseudo model* instead of "concept model."

For testing whether the conjunction of a set of concepts $\{\mathsf{C}_1, \ldots, \mathsf{C}_\mathsf{n}\}$ is satisfiable, we present and analyze a technique called *deep model merging* that generalizes the original model merging approach [5] in two ways: (1) we extend the model merging technique to the logic $\mathcal{ALCNH}_{R^+}$, i.e. this technique also deals with number restrictions; (2) we introduce *deep* pseudo models for concepts that are *recursively* traversed and checked for possible clashes.

Let A be a concept name, R a role name, and C a concept. The consistency of the initial ABox $\mathcal{A} = \{\mathsf{a}:\mathsf{C}\}$ is tested. If $\mathcal{A}$ is inconsistent, the *pseudo model*[4] of C is defined as $\bot$. If $\mathcal{A}$ is consistent, then there exists a non-empty set of completions $\mathcal{C}$. A completion $\mathcal{A}' \in \mathcal{C}$ is selected and a pmodel M for a concept C

[3] In our case the rules R$\sqcup$ and R$\exists_{\leq n}$.

[4] For brevity a pseudo model is also called a *pmodel*.

is defined as the tuple $\langle M^{\mathsf{A}}, M^{\neg\mathsf{A}}, M^{\exists}, M^{\forall}\rangle$ of concept sets using the following definitions.

$$M^{\mathsf{A}} = \{\mathsf{A} \,|\, \mathsf{a}:\mathsf{A} \in \mathcal{A}',\ \mathsf{A} \in C\}, \qquad M^{\neg\mathsf{A}} = \{\mathsf{A} \,|\, \mathsf{a}:\neg\mathsf{A} \in \mathcal{A}',\ \mathsf{A} \in C\}$$
$$M^{\exists} = \{\exists\,\mathsf{R}\,.\,\mathsf{C} \,|\, \mathsf{a}:\exists\,\mathsf{R}\,.\,\mathsf{C} \in \mathcal{A}'\} \cup \{\exists_{\geq n}\,\mathsf{R} \,|\, \mathsf{a}:\exists_{\geq n}\,\mathsf{R} \in \mathcal{A}'\}$$
$$M^{\forall} = \{\forall\,\mathsf{R}\,.\,\mathsf{C} \,|\, \mathsf{a}:\forall\,\mathsf{R}\,.\,\mathsf{C} \in \mathcal{A}'\} \cup \{\exists_{\leq n}\,\mathsf{R} \,|\, \mathsf{a}:\exists_{\leq n}\,\mathsf{R} \in \mathcal{A}'\} \cup \{\exists\,\mathsf{R}\,.\,\mathsf{C} \,|\, \mathsf{a}:\exists\,\mathsf{R}\,.\,\mathsf{C} \in \mathcal{A}',\ \mathsf{R} \in F\}$$

Note that pmodels are based on complete ABoxes. In contrast to the theoretical calculus presented above, model merging deals directly with features instead of representing them with at most restrictions. Therefore concept exists restrictions mentioning features are also included in the sets $M^{\forall}$ of pmodels. This guarantees that a possible "feature interaction" between pmodels is detected.

Procedure 1 mergable(MS, VM, $D?$)

```
1: if $MS = \emptyset \vee MS \in VM$ then
2:    return $true$
3: else if $\bot \in MS \vee \neg$atoms_mergable($MS$) then
4:    return $false$
5: else
6:    for all $M \in MS$ do
7:       for all $\mathsf{C} \in M^{\exists}$ do
8:          if critical_at_most($\mathsf{C}, M, MS$) then
9:             return $false$
10:         else
11:            $MS' \leftarrow$ collect_successor_pmodels($\mathsf{C}, MS$)
12:            if $(\neg D? \wedge MS' \neq \emptyset) \vee \neg$mergable($MS'$, $VM \cup \{\mathsf{MS}\}$, $D?$) then
13:               return $false$
14: return $true$
```

The procedure **mergable** shown in Procedure 1 implements the flat and deep model merging test. The test has to discover potential clashes which might occur if all pmodels in MS are merged, i.e. their corresponding concepts are conjunctively combined. The test starts with a set of pmodels MS, an empty set of visited pmodel sets VM, and a parameter $D?$ controlling whether the deep or flat mode (see below) of **mergable** will be used. The test recursively traverses the pmodel structures. In case of a potential clash, **mergable** terminates and returns *false*, otherwise it continues its traversal and returns *true* if no potential clash can be discovered. Testing whether the actual pmodel set MS is already a member of the set VM (line 1) is necessary to ensure termination (in analogy to blocking an individual) for the deep mode. A potential primitive clash is checked in line 3. If no primitive clash is possible for the "root individual" of the pmodel, it is tested whether a clash might be caused by interacting concept exists, concept value, at least, and at most restrictions for the same successor individuals. Two nested loops (lines 6-13) check for every pmodel $M \in MS$ and every concept in the set $M^{\exists}$ whether an at most restriction might be violated by the other pmodels (line 8). If this is not the case,[5] the set MS' of "R-successor pmodels" is computed (line 11). If the flat mode is enabled and $MS' \neq \emptyset$, this indicates a potential interaction via the role R and **mergable** returns *false* (lines 12-13). If the deep mode is enabled, **mergable** continues and traverses the pmodels in MS'. Observe that the procedure **mergable** is sound but not complete, i.e. even if **mergable** returns *false* for a pmodel set the corresponding concept conjunction can be satisfiable.

The procedure **atoms_mergable** tests for a possible primitive clash between pairs of pmodels. It is applied to a set of pmodels MS and returns *false* if there exists a pair $\{M_1, M_2\} \subseteq MS$ with $(M_1^{\mathsf{A}} \cap M_2^{\neg\mathsf{A}}) \neq \emptyset$ or $(M_1^{\neg\mathsf{A}} \cap M_2^{\mathsf{A}}) \neq \emptyset$. Otherwise it returns *true*.

The procedure **critical_at_most** checks for a potential number restriction clash in a set of pmodels and tries to avoid positive answers which are too conservative. It is applied to a concept C of the form $\exists\,\mathsf{S}\,.\,\mathsf{D}$ or $\exists_{\geq n}\,\mathsf{S}$, the current pmodel M and a set of pmodels $MS = \{M_1, \ldots, M_k\}$. Loosely speaking, it computes the maximal number of potential S-successors and returns *true* if this number exceeds the applicable at most bound m. More precisely, **critical_at_most** returns *true* if there exists a pmodel $M' \in (MS \backslash M)$ and a role $\mathsf{R} \in \mathsf{S}^{\uparrow}$ with $\exists_{\leq m}\,\mathsf{R} \in M'^{\forall}$ such that $\sum_{\mathsf{E} \in N} \mathrm{num}(\mathsf{E}, RS) > m$, $N = \cup_{i \in 1..k} M_i^{\exists}$, $RS = \mathsf{S}^{\uparrow} \cap \mathsf{R}^{\downarrow}$. In all other cases **critical_at_most** returns *false*. The function $\mathrm{num}(\mathsf{E}, RS)$ returns 1 for concepts of the form $\mathsf{E} = \exists\,\mathsf{R}'\,.\,\mathsf{D}$ and n for $\mathsf{E} = \exists_{\geq n}\,\mathsf{R}'$, if $\mathsf{R}' \in RS$, and 0 otherwise.

The procedure **collect_successor_pmodels** is applied to a concept C of the form $\exists\,\mathsf{S}\,.\,\mathsf{D}$ or $\exists_{\geq n}\,\mathsf{S}$ and a set of pmodels MS. It computes the set Q containing all S-successor pmodels (by considering (transitive) superroles of S). We define $Q_{aux} = \{\mathsf{D}\}$ if $\mathsf{C} = \exists\,\mathsf{S}\,.\,\mathsf{D}$ and $Q_{aux} = \emptyset$ otherwise. Observe that $\exists\,\mathsf{R}\,.\,\mathsf{E} \in M^{\forall}$ implies that R is a feature. The procedure **collect_successor_pmodels** returns the pmodel set $\{M_C \mid C \in Q\}$.

$$Q = Q_{aux} \cup \{\mathsf{E} \mid \exists\, M \in MS, \mathsf{R} \in \mathsf{S}^{\uparrow} : (\forall\,\mathsf{R}\,.\,\mathsf{E} \in M^{\forall} \vee \exists\,\mathsf{R}\,.\,\mathsf{E} \in M^{\forall})\} \cup$$
$$\{\forall\,\mathsf{T}\,.\,\mathsf{E} \mid \exists\, M \in MS, \mathsf{R} \in \mathsf{S}^{\uparrow}, \mathsf{T} \in T \cap \mathsf{S}^{\uparrow} \cap \mathsf{R}^{\downarrow} : \forall\,\mathsf{R}\,.\,\mathsf{E} \in M^{\forall}\}$$

Note that **mergable** depends on the clash triggers of the particular tableaux calculus chosen since it has to detect potential clashes in a set of pmodels. The structure and composition of the completion rules might vary as long as the clash triggers do not change and the calculus remains sound and complete.

Proposition 1 (Soundness of mergable). *Let $D?$ have either the value* true *or* false*, $CS = \{\mathsf{C}_1, \ldots, \mathsf{C}_n\}$, $M_{C_i} = \mathrm{get\_pmodel}(\mathsf{C}_i)$, and $PM = \{M_{C_i} \mid \mathsf{i} \in 1..\mathsf{n}\}$. If the procedure call **mergable**$(PM, \emptyset, D?)$ returns* true*, the concept $\mathsf{C}_1 \sqcap \ldots \sqcap \mathsf{C}_n$ is satisfiable.*

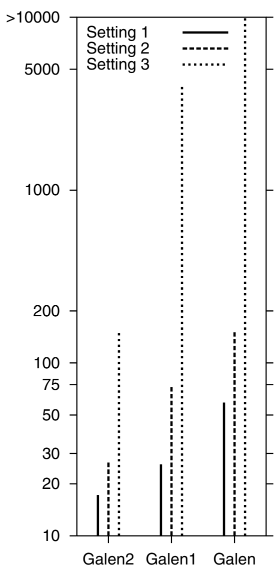
Proof. This is proven by contradiction and induction. Let us assume that the call **mergable**$(PM, \emptyset, D?)$ returns *true* but the initial ABox $\mathcal{A} = \{\mathsf{a} : (\mathsf{C}_1 \sqcap \ldots \sqcap \mathsf{C}_n)\}$ is inconsistent, i.e. there exists no completion of $\mathcal{A}$. Every concept C_i must be

[5] In the following let us assume that the concept C mentions a role name R.

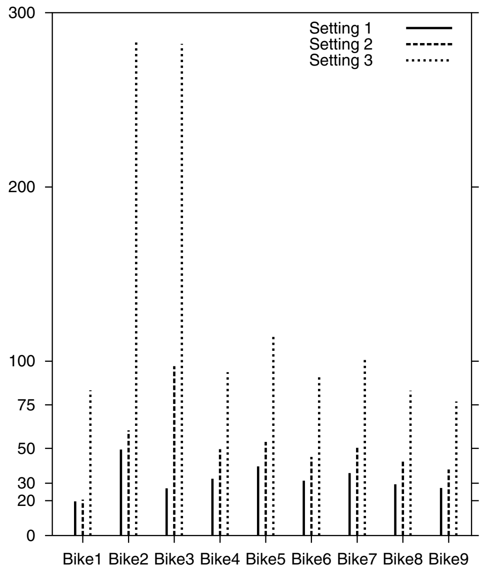
satisfiable, otherwise we would have $\bot \in PM$ and **mergable** would return *false* due to line 3 in Procedure 1. Let us assume a finite set $\mathcal{C}$ containing all contradictory ABoxes encountered during the consistency test of $\mathcal{A}$. Without loss of generality we can select an arbitrary $\mathcal{A}' \in \mathcal{C}$ and make a case analysis of its possible clash culprits.

1. We have a primitive clash for the "root" individual a, i.e. $\{\mathsf{a}:\mathsf{D}, \mathsf{a}:\neg\mathsf{D}\} \subseteq \mathcal{A}'$. Thus, $\mathsf{a}:\mathsf{D}$ and $\mathsf{a}:\neg\mathsf{D}$ have not been propagated to a via role assertions and there have to exist $\mathsf{C_i}, \mathsf{C_j} \in CS$, $\mathsf{i} \neq \mathsf{j}$ such that $\mathsf{a}:\mathsf{D}$ ($\mathsf{a}:\neg\mathsf{D}$) is derived from $\mathsf{a}:\mathsf{C_i}$ ($\mathsf{a}:\mathsf{C_j}$) due to the satisfiability of the concepts $\mathsf{C_i}$, $\mathsf{i} \in 1..\mathsf{n}$. It holds for the associated pmodels $M_{C_i}, M_{C_j} \in PM$ that $\mathsf{D} \in M^{\mathsf{A}}_{C_i} \cap M^{\neg\mathsf{A}}_{C_j}$. However, due to our assumption the call of $\mathbf{mergable}(PM, \emptyset, D?)$ returned *true*. This is a contradiction since **mergable** called **atoms_mergable** with PM (line 3 in Procedure 1) which returned *false* since $\mathsf{D} \in M^{\mathsf{A}}_{C_i} \cap M^{\neg\mathsf{A}}_{C_j}$.
2. A number restriction clash in $\mathcal{A}'$ is detected for a, i.e. $\mathsf{a}:\exists_{\leq m}\,\mathsf{R} \in \mathcal{A}'$ and there exist $l > m$ distinct R-successors of a.[6] These successors can only be derived from assertions of the form $\mathsf{a}:\exists\,\mathsf{S_j}\,.\,\mathsf{E_j}$ or $\mathsf{a}:\exists_{\geq n_j}\,\mathsf{S_j}$ with $\mathsf{S_j} \in \mathsf{R}^{\downarrow}$, $\mathsf{j} \in 1..\mathsf{p}$. The concepts $\mathsf{C_i} \in CS$, $\mathsf{i} \in 1..\mathsf{n}$ are satisfiable and there has to exist a subset $CS' \subseteq CS$ such that $\exists_{\leq m}\,\mathsf{R} \in \cup_{C \in CS'} M^{\forall}_{C}$ and $\sum_{\mathsf{E}' \in N} \mathrm{num}(\mathsf{E}', RS) \geq l$, $N = \cup_{C \in CS'} M^{\exists}_{C}$, $RS = (\cup_{\mathsf{j} \in 1..\mathsf{p}} \mathsf{S_j}^{\uparrow}) \cap \mathsf{R}^{\downarrow}$. However, due to our assumption the call of $\mathbf{mergable}(PM, \emptyset, D?)$ returned *true*. This is a contradiction since there exists a pmodel M'_C, $\mathsf{C} \in CS'$ and a concept $\mathsf{E}' \in {M'}^{\exists}_{C}$ such that **mergable** called $\mathbf{critical\_at\_most}(\mathsf{E}', M'_C, PM)$ (lines 6-8 in Procedure 1) which returned *true* since $\sum_{\mathsf{E}' \in N} \mathrm{num}(\mathsf{E}', RS) \geq l > m$.
3. Let the individual $\mathsf{a_n}$ be a successor of $\mathsf{a_0}$ via a chain of role assertions $(\mathsf{a_0}, \mathsf{a_1}):\mathsf{R_1}, \ldots, (\mathsf{a_{n-1}}, \mathsf{a_n}):\mathsf{R_n}$, $\mathsf{n} > 0$ and we now assume that a clash for $\mathsf{a_n}$ is discovered.
   (a) In case of a primitive clash we have $\{\mathsf{a_n}:\mathsf{D}, \mathsf{a_n}:\neg\mathsf{D}\} \subseteq \mathcal{A}'$. Without loss of generality we may assume that the clash culprits can only be derived from assertions of the form $\mathsf{a_{n-1}}:\exists_{\geq m}\,\mathsf{R_n}$ or $\mathsf{a_{n-1}}:\exists\,\mathsf{R_n}\,.\,\mathsf{E_1}$ in combination with $\mathsf{a_{n-1}}:\exists\,\mathsf{S}'\,.\,\mathsf{E_2}$ (if $\mathsf{R_n}$ and $\mathsf{S}' \in \mathsf{R_n}^{\uparrow}$ are features), and/or $\mathsf{a_{n-1}}:\forall\,\mathsf{S}''\,.\,\mathsf{E_3}$ with $\mathsf{S}'' \in \mathsf{R_n}^{\uparrow}$. Due to the clash there exists a pair $\mathsf{E}', \mathsf{E}'' \subseteq \{\mathsf{E_1}, \mathsf{E_2}, \mathsf{E_3}\}$ with $\mathsf{D} \in M^{\mathsf{A}}_{E'} \cap M^{\neg\mathsf{A}}_{E''}$. Each role assertion in the chain between $\mathsf{a_0}$ and $\mathsf{a_{n-1}}$ can only be derived from assertions of the form $\mathsf{a_{k-1}}:\exists\,\mathsf{R_k}\,.\,\mathsf{E_k}$ or $\mathsf{a_{k-1}}:\exists_{\geq m_k}\,\mathsf{R_k}$ with $\mathsf{k} \in 1..\mathsf{n}-1$. The call graph of $\mathbf{mergable}(PM, \emptyset, D?)$ contains a chain of calls resembling the chain of role assertions. By induction on the call graph we know that the node resembling $\mathsf{a_{n-1}}$ of this call graph chain contains the call $\mathbf{mergable}(PM', VM', true)$ such that $\{M_{E'}, M_{E''}\} \subseteq PM'$ and **atoms_mergable** has been called with a set MS' and $\{M_{E'}, M_{E''}\} \subseteq MS'$. The call of **atoms_mergable** has returned *false* since $\mathsf{D} \in M^{\mathsf{A}}_{E'} \cap M^{\neg\mathsf{A}}_{E''}$. This contradicts our assumption that $\mathbf{mergable}(PM, \emptyset, D?)$ returned *true*.
   (b) In case of a number restriction clash we can argue in an analogous way to case 2 and 3a. Again, we have a chain of role assertions where a number restriction clash is detected for the last individual of the chain.

[6] Due to our syntax restriction, the elements of $\mathsf{R}^{\downarrow}$ are not transitive.



(a) Galen TBoxes



(b) Bike TBoxes

Fig. 2. Evaluation of model merging techniques (runtime in seconds, 3 runs for each TBox, left-right order corresponds to top-bottom order in the legend).

It exists a corresponding call graph chain where by induction the last call of **mergable** called **critical_at_most** with a set of pmodels for which **critical_at_most** returned *true*. This contradicts the assumption that **mergable**($PM, \emptyset, D?$) returned *true*. □

It is easy to see that this proof also holds if the value of $D?$ is *false* since the "flat mode" is more conservative than the "deep" one, i.e. it will always return *false* instead of possibly *true* if the set of collected pmodels M' is not empty (line 12 in Procedure 1).

The advantage of the deep vs. the flat mode of the model merging technique is demonstrated by empirical tests using a set of "quasi-standard" application TBoxes [7, 6, 2]. Figure 2 shows the runtimes for computing the subsumption lattice of these TBoxes. Each TBox is iteratively classified using three different parameter settings. The first setting has the deep mode of model merging enabled, the second one has the deep mode of model merging disabled but the flat mode still enabled, and the third one has model merging completely disabled. The comparison between setting one and two indicates a speed up in runtimes of a factor $1.5 - 2$ if the deep mode is enabled. The result for setting three clearly demonstrate the principal advantage of model merging.

The principal advantage of the deep vs. the flat model merging mode is due to the following characteristics. If the flat model merging test is (recursively) applied during tableaux expansion and repeatedly returns *false* because of interacting value and exists restrictions, this test might be too conservative. This effect is illustrated by an example: The deep model merging test starts with the pmodels $\langle \emptyset, \emptyset, \{\exists\, \mathsf{R} \,.\, \exists\, \mathsf{S} \,.\, \mathsf{C}\}, \emptyset \rangle$ and $\langle \emptyset, \emptyset, \emptyset, \{\forall\, \mathsf{R} \,.\, \forall\, \mathsf{S} \,.\, \mathsf{D}\} \rangle$. Due to interaction on the role R, the test is recursively applied to the pmodels $\langle \emptyset, \emptyset, \{\exists\, \mathsf{S} \,.\, \mathsf{C}\}, \emptyset \rangle$ and $\langle \emptyset, \emptyset, \emptyset, \{\forall\, \mathsf{S} \,.\, \mathsf{D}\} \rangle$. Eventually, the deep model merging test succeeds with the pmodels $\langle \{\mathsf{C}\}, \emptyset, \emptyset, \emptyset \rangle$ and $\langle \{\mathsf{D}\}, \emptyset, \emptyset, \emptyset \rangle$ and returns true. This is in contrast to the flat mode where in this example no tableaux tests are avoided and the runtime for the model merging tests is wasted.

The next section describes how model merging can be utilized for obtaining a dramatic speed up of ABox reasoning.

3 Flat Models for ABox Reasoning in $\mathcal{ALCNH}_{R^+}$

Computing the *direct types* of an individual a (i.e. the set of the most specific concepts from C of which an individual a is an instance) is called *realization* of a. For instance, in order to compute the direct types of a for a given subsumption lattice of the concepts $\mathsf{D_1}, \ldots, \mathsf{D_n}$, a sequence of ABox consistency tests for $\mathcal{A}_{\mathsf{D_i}} = \mathcal{A} \cup \{\mathsf{a} : \neg \mathsf{D_i}\}$ might be required. However, individuals are usually members of only a small number of concepts and the ABoxes $\mathcal{A}_{\mathsf{D_i}}$ are proven as consistent in most cases. The basic idea is to design a cheap but *sound* model merging test for the focused individual a and the concept terms $\neg \mathsf{D_i}$ without explicitly considering role assertions and concept assertions for all the other individuals mentioned in $\mathcal{A}$. These "interactions" are reflected in the "individual pseudo model" of a. This is the motivation for devising the novel *individual model merging* technique.

A pseudo model for an individual a mentioned in a consistent initial ABox $\mathcal{A}$ w.r.t. a TBox $\mathcal{T}$ is defined as follows. Since $\mathcal{A}$ is consistent, there exists a set of completions $\mathcal{C}$ of $\mathcal{A}$. Let $\mathcal{A}' \in \mathcal{C}$. An *individual pseudo model* M for an individual a in $\mathcal{A}$ is defined as the tuple $\langle M^{\mathsf{A}}, M^{\neg \mathsf{A}}, M^{\exists}, M^{\forall} \rangle$ w.r.t. $\mathcal{A}'$ and $\mathcal{A}$ using the same definitions from the previous section for the components $M^{\mathsf{A}}, M^{\neg \mathsf{A}}, M^{\forall}$ and the following definition.

$$M^{\exists} = \{\exists\, \mathsf{R} \,.\, \mathsf{C} \mid \mathsf{a} : \exists\, \mathsf{R} \,.\, \mathsf{C} \in \mathcal{A}'\} \cup \{\exists_{\geq n}\, \mathsf{R} \mid \mathsf{a} : \exists_{\geq n}\, \mathsf{R} \in \mathcal{A}'\} \cup \{\exists_{\geq 1}\, \mathsf{R} \mid (\mathsf{a}, \mathsf{b}) : \mathsf{R} \in \mathcal{A}\}$$

Note the distinction between the initial ABox $\mathcal{A}$ and its completion $\mathcal{A}'$. Whenever a role assertion exists, which specifies a role successor for the individual a in the initial ABox, a corresponding at least restriction is added to the set $M^{\exists}$. This is based on the rationale that the cached pmodel of a cannot refer to individual names. However, it is sufficient to reflect a role assertion $(\mathsf{a}, \mathsf{b}) : \mathsf{R} \in \mathcal{A}$ by adding a corresponding at least restriction to $M^{\exists}$. This guarantees that possible interactions via the role R are detected. Note that individual model merging is only defined for the *flat* mode of model merging.

Proposition 2 (Soundness of individual_model_merging). *Let M_a be the pmodel of an individual* a *mentioned in a consistent initial ABox $\mathcal{A}$, $M_{\neg C}$ be the pmodel of a satisfiable concept* $\neg \mathsf{C}$*, and $PM = \{M_a, M_{\neg C}\}$. If the procedure call* ***mergable***$(PM, \emptyset, \mathit{false})$ *returns* true*, the ABox $\mathcal{A} \cup \{\mathsf{a} : \neg \mathsf{C}\}$ is consistent, i.e.* a *is not an instance of* C.

Proof. This is proven by contradiction. Let us assume that the procedure call **mergable**$(\{M_a, M_{\neg C}\}, \emptyset, \mathit{false})$ returns *true* but the ABox $\mathcal{A}' = \mathcal{A} \cup \{\mathsf{a} : \neg\mathsf{C}\}$ is inconsistent, i.e. there exists no completion of $\mathcal{A}'$. Let us assume a finite set $\mathcal{C}$ containing all contradictory ABoxes encountered during the consistency test of $\mathcal{A}'$. Without loss of generality we can select an arbitrary $\mathcal{A}'' \in \mathcal{C}$ and make a case analysis of its possible clash culprits.

1. In case of a primitive clash for a we have $\{\mathsf{a} : \mathsf{D}, \mathsf{a} : \neg\mathsf{D}\} \subseteq \mathcal{A}''$. Since $\mathcal{A}$ is consistent and the concept $\neg\mathsf{C}$ cannot indirectly refer to the old individual a via a role chain, we know that either $\mathsf{a} : \mathsf{D}$ or $\mathsf{a} : \neg\mathsf{D}$ must be derived from $\mathsf{a} : \neg\mathsf{C}$ and we have $\mathsf{D} \in (M_a^{\mathsf{A}} \cap M_{\neg C}^{\neg\mathsf{A}}) \cup (M_a^{\neg\mathsf{A}} \cap M_{\neg C}^{\mathsf{A}})$. This contradicts the assumption that the call **mergable**$(\{M_a, M_{\neg C}\}, \emptyset, \mathit{false})$ returned *true* since **mergable** called **atoms_mergable**$(\{M_a, M_{\neg C}\})$ which returned *false* (line 3 in Procedure 1) since $\mathsf{D} \in (M_a^{\mathsf{A}} \cap M_{\neg C}^{\neg\mathsf{A}}) \cup (M_a^{\neg\mathsf{A}} \cap M_{\neg C}^{\mathsf{A}})$.
2. A number restriction clash in $\mathcal{A}''$ is detected for a, i.e. $\mathsf{a} : \exists_{\leq m} \mathsf{R} \in \mathcal{A}''$ and there exist $l > m$ distinct R-successors of a in $\mathcal{A}''$. This implies that the set $N = M_a^{\exists} \cup M_{\neg C}^{\exists}$ contains concepts of the form $\exists\, \mathsf{S_j} \,.\, \mathsf{E_j}$ or $\exists_{\geq n_j} \mathsf{S_j}$,[7] $\mathsf{S_j} \in \mathsf{R}^{\downarrow}$, $\mathsf{j} \in 1..\mathsf{k}$, such that $\sum_{\mathsf{E}' \in N} \mathrm{num}(\mathsf{E}', \mathit{RS}) \geq l$, $\mathit{RS} = (\cup_{\mathsf{j} \in 1..\mathsf{k}} \mathsf{S_j}^{\uparrow}) \cap \mathsf{R}^{\downarrow}$. This contradicts the assumption that **mergable**$(\{M_a, M_{\neg C}\}, \emptyset, \mathit{false})$ returned *true* since mergable called critical_at_most (lines 6-8 in Procedure 1) which returned *true* since $\sum_{\mathsf{E}' \in N} \mathrm{num}(\mathsf{E}', \mathit{RS}) \geq l > m$.
3. A clash is detected for an individual b in $\mathcal{A}''$ that is distinct to a. Since $\mathcal{A}$ is consistent the individual b must be a successor of a via a chain of role assertions $(\mathsf{a}, \mathsf{b_1}) : \mathsf{R_1}, \ldots, (\mathsf{b_n}, \mathsf{b}) : \mathsf{R_{n+1}}, \mathsf{n} \geq 0$, and one of the clash culprits must be derived from the newly added assertion $\mathsf{a} : \neg\mathsf{C}$ and propagated to b via the role assertion chain originating from a with $(\mathsf{a}, \mathsf{b_1}) : \mathsf{R_1}$. Since $\neg\mathsf{C}$ is satisfiable and $\mathcal{A}$ is consistent we have an "interaction" via the role or feature $\mathsf{R_1}$. This implies for the associated pmodels $M_a, M_{\neg C}$ that $(M_a^{\exists} \cap M_{\neg C}^{\forall}) \cup (M_a^{\forall} \cap M_{\neg C}^{\exists}) \neq \emptyset$. This contradicts the assumption that **mergable**$(\{M_a, M_{\neg C}\}, \emptyset, \mathit{false})$ returned *true* since **mergable** eventually called **collect_successor_pmodels** for $M_a, M_{\neg C}$ which returned a non-empty set (line 11 in Procedure 1). □

The performance gain by the individual model merging technique is empirically evaluated using a set of five ABoxes containing between 15 and 25 individuals. Each of these ABoxes is realized w.r.t. the application TBoxes Bike7-9 derived from a bike configuration task. The TBoxes especially vary on the degree of explicit disjointness declarations between atomic concepts. Figure 3 shows the runtimes for the realization of the ABoxes 1-5. Each ABox is realized with two different parameter settings. The first setting has the individual model merging technique enabled, the second one has it disabled. The comparison between both settings reveals a speed gain of at least one order of magnitude if the individual model merging technique is used. Note the use of a logarithmic scale.

[7] Any role assertion of the form $(\mathsf{a}, \mathsf{b}) : \mathsf{R} \in \mathcal{A}$ implies that $\exists_{\geq 1} \mathsf{R} \in M_a^{\exists}$. This takes care of implied at least restrictions due to the UNA for old individuals.

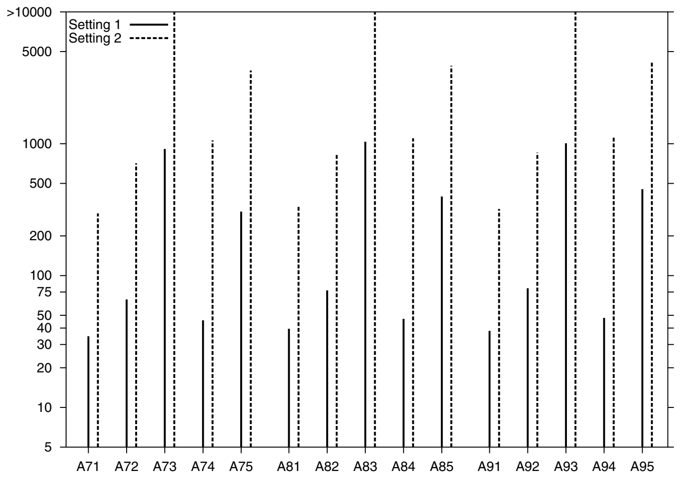


Fig. 3. Bike ABoxes: Evaluation of model merging techniques (runtime in seconds, 2 runs for each ABox, left-right order corresponds to top-bottom order in the legend).

4 Pseudo Models for Reasoning with Concrete Domains

The requirements derived from practical applications of DLs ask for more expressiveness w.r.t. reasoning about objects from other domains (so-called concrete domains, e.g. for the real numbers). Thus, in [4] the logic $\mathcal{ALCNH}_{R^+}$ is extended with a restricted form of reasoning about concrete domains. However, the classification of non-trivial TBoxes is only feasible, if the model merging technique can be applied. Therefore, we extend the model merging technique to the basic DL with concrete domains, the language $\mathcal{ALC}(\mathcal{D})$ [1]. We conjecture that the results from this approach can be directly transferred to the logic presented in [4]. First, we have to briefly introduce $\mathcal{ALC}(\mathcal{D})$.

4.1 The Language $\mathcal{ALC}(\mathcal{D})$

A *concrete domain* $\mathcal{D}$ is a pair $(\Delta_\mathcal{D}, \Phi_\mathcal{D})$, where $\Delta_\mathcal{D}$ is a set called the domain, and $\Phi_\mathcal{D}$ is a set of predicate names. Each predicate name $\mathsf{P}_\mathcal{D}$ from $\Phi_\mathcal{D}$ is associated with an arity n and an n-ary predicate $\mathsf{P}_\mathcal{D} \subseteq \Delta_\mathcal{D}^n$. A concrete domain $\mathcal{D}$ is called *admissible* iff (1) the set of predicate names $\Phi_\mathcal{D}$ is closed under negation and $\Phi_\mathcal{D}$ contains a name $\top_\mathcal{D}$ for $\Delta_\mathcal{D}$; (2) the satisfiability problem $\mathsf{P}_1^{\mathsf{n}_1}(\mathsf{x}_{11}, \ldots, \mathsf{x}_{1\mathsf{n}_1}) \wedge \ldots \wedge \mathsf{P}_\mathsf{m}^{\mathsf{n}_\mathsf{m}}(\mathsf{x}_{\mathsf{m}1}, \ldots, \mathsf{x}_{\mathsf{m}\mathsf{n}_\mathsf{m}})$ is decidable (m is finite, $\mathsf{P}_\mathsf{i}^{\mathsf{n}_\mathsf{i}} \in \Phi_\mathcal{D}$, and x_{jk} is a name for an object from $\Delta_\mathcal{D}$).

Let S and F ($R = S \cup F$) be disjoint sets of role and feature names, respectively. A composition of features (written $\mathsf{F}_1 \cdots \mathsf{F}_\mathsf{n}$) is called a *feature chain*. A simple feature is a feature chain of length 1. Let C be a set of concept names which is disjoint from R. Any element of C is a *concept term*. If C and D are concept terms, $\mathsf{R} \in R$ is a role or feature name, $\mathsf{P} \in \Phi_\mathcal{D}$ is a predicate name from an admissible concrete domain, $\mathsf{u_i}$'s are feature chains, then the following expressions are also concept terms: $\mathsf{C} \sqcap \mathsf{D}$, $\mathsf{C} \sqcup \mathsf{D}$, $\neg\mathsf{C}$, $\forall\, \mathsf{R} \,.\, \mathsf{C}$, $\exists\, \mathsf{R} \,.\, \mathsf{C}$, $\exists\, \mathsf{u_1}, \ldots, \mathsf{u_n} \,.\, \mathsf{P}$. A concept term of the last kind is called *predicate exists restriction*.

An *interpretation* $\mathcal{I}_\mathcal{D} = (\Delta^\mathcal{I}, \Delta^\mathcal{D}, \cdot^\mathcal{I})$ consists of a set $\Delta^\mathcal{I}$ (the abstract domain), a set $\Delta^\mathcal{D}$ (the domain of an admissible 'concrete domain' $\mathcal{D}$) and an interpretation function $\cdot^\mathcal{I}$. Besides for feature and predicate names the interpretation function is defined as in Figure 1a. The function maps each feature name F from F to a partial function $\mathsf{F}^\mathcal{I}$ from $\Delta^\mathcal{I}$ to $\Delta^\mathcal{I} \cup \Delta^\mathcal{D}$, and each predicate name P from $\Phi_\mathcal{D}$ with arity n to a subset $\mathsf{P}^\mathcal{I}$ of $\Delta^n_\mathcal{D}$. For a feature chain $\mathsf{u} = \mathsf{F_1} \cdots \mathsf{F_n}$, $\mathsf{u}^\mathcal{I}$ denotes the composition ${\mathsf{F_1}}^\mathcal{I} \circ \cdots \circ {\mathsf{F_n}}^\mathcal{I}$ of partial functions ${\mathsf{F_1}}^\mathcal{I}, \ldots, {\mathsf{F_n}}^\mathcal{I}$. Let $\mathsf{u_1}, \ldots, \mathsf{u_n}$ be feature chains and let P be a predicate name. Then, the interpretation function can be extended to concept terms as in Figure 1a. The semantics for the predicate exists restrictions is given by:

$$(\exists\, \mathsf{u_1}, \ldots, \mathsf{u_n} \,.\, \mathsf{P})^\mathcal{I} := \{\, a \in \Delta^\mathcal{I} \mid \exists x_1, \ldots, x_n \in \Delta^\mathcal{D} : (a, x_1) \in {\mathsf{u_1}}^\mathcal{I}, \ldots, (a, x_n) \in {\mathsf{u_n}}^\mathcal{I}, \\ (x_1, \ldots, x_n) \in \mathsf{P}^\mathcal{I} \}$$

Note that in a concept term elements of $\Delta^\mathcal{D}$ can be used only as feature fillers.

A TBox $\mathcal{T}$ is a finite set of non-cyclic axioms of the form $\mathsf{A} \sqsubseteq \mathsf{D}$ or $\mathsf{A} \doteq \mathsf{D}$ where A must be a concept name. An interpretation $\mathcal{I}$ is a *model* of a TBox $\mathcal{T}$ iff it satisfies $\mathsf{A}^\mathcal{I} \subseteq \mathsf{D}^\mathcal{I}$ ($\mathsf{A}^\mathcal{I} = \mathsf{D}^\mathcal{I}$) for all $\mathsf{A} \sqsubseteq \mathsf{D}$ ($\mathsf{A} \doteq \mathsf{D}$) in $\mathcal{T}$.

An *ABox* $\mathcal{A}$ is a finite set of assertional axioms which are defined as follows: Let O be a set of individual names and let X be a set of names for concrete objects ($X \cap O = \emptyset$). If C is a concept term, $\mathsf{R} \in R$, $\mathsf{F} \in F$, $\mathsf{a}, \mathsf{b} \in O$ and $\mathsf{x}, \mathsf{x_1}, \ldots, \mathsf{x_n} \in X$, then the following expressions are *assertional axioms*: $\mathsf{a} : \mathsf{C}$, $(\mathsf{a}, \mathsf{b}) : \mathsf{R}$, $(\mathsf{a}, \mathsf{x}) : \mathsf{F}$ and $(\mathsf{x_1}, \ldots, \mathsf{x_n}) : \mathsf{P}$.

The interpretation function additionally maps every individual name from O to a single element of $\Delta^\mathcal{I}$ and names for concrete objects from X are mapped to elements of $\Delta^\mathcal{D}$. (The UNA does not necessarily hold in $\mathcal{ALC}(\mathcal{D})$.) An interpretation satisfies an assertional axiom $\mathsf{a} : \mathsf{C}$ iff $\mathsf{a}^\mathcal{I} \in \mathsf{C}^\mathcal{I}$, $(\mathsf{a}, \mathsf{b}) : \mathsf{R}$ iff $(\mathsf{a}^\mathcal{I}, \mathsf{b}^\mathcal{I}) \in \mathsf{R}^\mathcal{I}$, $(\mathsf{a}, \mathsf{x}) : \mathsf{F}$ iff $(\mathsf{a}^\mathcal{I}, \mathsf{x}^\mathcal{I}) \in \mathsf{F}^\mathcal{I}$, and $(\mathsf{x_1}, \ldots, \mathsf{x_n}) : \mathsf{P}$ iff $({\mathsf{x_1}}^\mathcal{I}, \ldots, {\mathsf{x_n}}^\mathcal{I}) \in \mathsf{P}^\mathcal{I}$. An interpretation $\mathcal{I}$ is a *model* of an ABox $\mathcal{A}$ w.r.t. a TBox $\mathcal{T}$ iff it is a model of $\mathcal{T}$ and furthermore satisfies all assertional axioms in $\mathcal{A}$.

4.2 Pseudo Models for TBox Reasoning in $\mathcal{ALC}(\mathcal{D})$

By analogy to the previous sections, we assume a tableaux calculus which decides the ABox consistency problem for $\mathcal{ALC}(\mathcal{D})$ (see [1]). The clash triggers in this calculus are the primitive clash, two triggers for feature fillers with membership to both domains, and one clash trigger indicating inconsistencies between concrete domain objects.

In the following we assume the same naming conventions as used above. In order to obtain a *flat pseudo model* for a concept C the consistency of $\mathcal{A} = \{\mathsf{a} : \mathsf{C}\}$ is tested. If $\mathcal{A}$ is inconsistent, the *pseudo model* of C is defined as $\bot$. If $\mathcal{A}$ is consistent, then there exists a set of completions $\mathcal{C}$. A completion $\mathcal{A}' \in \mathcal{C}$ is selected and a pmodel M for a concept C is defined as the tuple

$\langle M^{\mathsf{A}}, M^{\neg\mathsf{A}}, M^{\exists}, M^{\forall}, M^{\exists\mathsf{F}}, M^{\forall\mathsf{F}}\rangle$ using the following definitions (let $\mathsf{u} = \mathsf{F_1} \cdots \mathsf{F_n}$ be a feature chain, then $\mathit{first}(\mathsf{u}) = \mathsf{F_1}$).

$$M^{\mathsf{A}} = \{\mathsf{A} \,|\, \mathsf{a}\!:\!\mathsf{A} \in \mathcal{A}'\}, \qquad M^{\neg\mathsf{A}} = \{\mathsf{A} \,|\, \mathsf{a}\!:\!\neg\mathsf{A} \in \mathcal{A}'\},$$
$$M^{\exists} = \{\mathsf{R} \,|\, \mathsf{a}\!:\!\exists\,\mathsf{R}\,.\,\mathsf{C} \in \mathcal{A}'\}, \qquad M^{\forall} = \{\mathsf{R} \,|\, \mathsf{a}\!:\!\forall\,\mathsf{R}\,.\,\mathsf{C} \in \mathcal{A}'\},$$
$$M^{\exists\mathsf{F}} = \{\mathsf{F} \,|\, \mathsf{a}\!:\!\exists\,\mathsf{F}\,.\,\mathsf{C} \in \mathcal{A}'\} \,\cup$$
$$\{\mathsf{F} \,|\, \mathsf{F} = \mathit{first}(\mathsf{u_j}), \mathsf{u_j} \text{ used in } \exists\,\mathsf{u_1},\ldots,\mathsf{u_n}\,.\,\mathsf{P},\ \mathsf{a}\!:\!\exists\,\mathsf{u_1},\ldots,\mathsf{u_n}\,.\,\mathsf{P} \in \mathcal{A}'\},$$
$$M^{\forall\mathsf{F}} = \{\mathsf{F} \,|\, \mathsf{a}\!:\!\forall\,\mathsf{F}\,.\,\mathsf{C} \in \mathcal{A}'\}$$

Note that sets from a flat pseudo model for an $\mathcal{ALC}(\mathcal{D})$ concept contain only concept, role, and/or feature names. In order to correctly deal with the semantics of features, the pmodel also contains separate sets $M^{\forall\mathsf{F}}$ and $M^{\exists\mathsf{F}}$. The set $M^{\exists\mathsf{F}}$ contains all feature names mentioned in exists restrictions and all feature names being first element of a feature chain in predicate exists restrictions, and the set $M^{\forall\mathsf{F}}$ contains all feature names mentioned in value restrictions.

The following procedure **$\mathcal{ALC}(\mathcal{D})$-mergable** implements the flat model merging test for $\mathcal{ALC}(\mathcal{D})$ for a given non-empty set of pmodels *MS*.

Procedure 2 $\mathcal{ALC}(\mathcal{D})$-mergable(*MS*)

```
if $\bot \in MS \vee \neg$atoms_mergable($MS$) then
  return false
else
  for all pairs $\{M_1, M_2\} \subseteq MS$ do
    if $(M_1^{\exists} \cap M_2^{\forall}) \neq \emptyset \vee (M_1^{\forall} \cap M_2^{\exists}) \neq \emptyset$ then
      return false
    else if $(M_1^{\exists\mathsf{F}} \cap M_2^{\forall\mathsf{F}}) \neq \emptyset \vee (M_1^{\forall\mathsf{F}} \cap M_2^{\exists\mathsf{F}}) \neq \emptyset \vee (M_1^{\exists\mathsf{F}} \cap M_2^{\exists\mathsf{F}}) \neq \emptyset$ then
      return false
return true
```

The idea of this test is to check for possible primitive clashes at the "root individual" of the pmodels in *MS* using **atoms_mergable**. Then the procedure **$\mathcal{ALC}(\mathcal{D})$-mergable** checks for possible references to the same direct role or feature filler by more than one pmodel in *MS*.

This easy, but conservative test handles, besides primitive clashes, the three $\mathcal{ALC}(\mathcal{D})$-specific clash triggers, because they can only appear at feature fillers. A proof for the soundness of **$\mathcal{ALC}(\mathcal{D})$-mergable** can therefore be easily adapted from the one given in Section 2. Due to lack of space, we cannot present the model merging technique for *deep* pseudo models which is described in [9] where this technique is also extended for other DLs with concrete domains. Full proofs for flat and deep model merging for $\mathcal{ALC}(\mathcal{D})$ can be found in [9].

5 Conclusion and Future Work

In this paper we have analyzed optimization techniques for TBox and ABox reasoning in the expressive description logic $\mathcal{ALCNH}_{R^+}$. These techniques exploit the traversal of flat and/or deep pmodels extracted from ABox consistency tests. A moderate speed gain using deep models for classification of concepts and a dramatic gain for realization of ABoxes is empirically demonstrated. The model merging technique has also been investigated for the logic $\mathcal{ALC}(\mathcal{D})$ with concrete domains. We conjecture that individual model merging for $\mathcal{ALC}(\mathcal{D})$ can be developed in analogy to Section 3. The model merging technique for $\mathcal{ALC}(\mathcal{D})$ is a prerequisite in order to apply model merging to $\mathcal{ALCNH}_{R^+}$ extended by concrete domains.

It is easy to see that an enhanced version of the individual model merging technique for $\mathcal{ALCNH}_{R^+}$ can be developed, which additionally exploits the use of deep models. This is immediately possible if only ABoxes containing no joins for role assertions are encountered. In case an ABox $\mathcal{A}$ contains a join (e.g. $\{(\mathsf{a},\mathsf{c}):\mathsf{R}, (\mathsf{b},\mathsf{c}):\mathsf{R}\} \subseteq \mathcal{A}$), one has to consider a graph-like instead of a tree-like traversal of pseudo models reflecting the dependencies caused by joins.

References

1. F. Baader and P. Hanschke. A scheme for integrating concrete domains into concept languages. In *Twelfth International Joint Conference on Artificial Intelligence, Darling Harbour, Sydney, Australia, Aug. 24-30, 1991*, pages 452–457, August 1991.
2. V. Haarslev and R. Möller. An empirical evaluation of optimization strategies for ABox reasoning in expressive description logics. In P. Lambrix et al., editor, *Proceedings of the International Workshop on Description Logics (DL'99), July 30 - August 1, 1999, Linköping, Sweden*, pages 115–119, June 1999.
3. V. Haarslev and R. Möller. Expressive ABox reasoning with number restrictions, role hierarchies, and transitively closed roles. In A.G. Cohn, F. Giunchiglia, and B. Selman, editors, *Proceedings of Seventh International Conference on Principles of Knowledge Representation and Reasoning (KR'2000), Breckenridge, Colorado, USA, April 11-15, 2000*, pages 273–284, April 2000.
4. V. Haarslev, R. Möller, and M. Wessel. The description logic $\mathcal{ALCNH}_{R^+}$ extended with concrete domains: A practically motivated approach. In *Proceedings of the International Joint Conference on Automated Reasoning, IJCAR'2001, June 18-23, 2001, Siena, Italy*, LNCS. Springer-Verlag, Berlin, June 2001.
5. I. Horrocks. *Optimising Tableaux Decision Procedures for Description Logics.* PhD thesis, University of Manchester, 1997.
6. I. Horrocks and P. Patel-Schneider. Optimising description logic subsumption. *Journal of Logic and Computation*, 9(3):267–293, June 1999.
7. I. Horrocks and P.F. Patel-Schneider. DL systems comparison. In *Proceedings of DL'98, International Workshop on Description Logics*, pages 55–57, Trento(Italy), 1998.
8. I. Horrocks, U. Sattler, and S. Tobies. Practical reasoning for expressive description logics. In H. Ganzinger, D. McAllester, and A. Voronkov, editors, *Proceedings of the 6th International Conference on Logic for Programming and Automated Reasoning (LPAR'99)*, number 1705 in Lecture Notes in Artificial Intelligence, pages 161–180. Springer-Verlag, September 1999.
9. A.-Y. Turhan. Optimization methods for the satisfiability test for description logics with concrete domains (in German). Master's thesis, University of Hamburg, Computer Science Department, April 2000.