Practical Reasoning in Racer with a Concrete Domain for Linear Inequations

Volker Haarslev
University of Hamburg
Computer Science Department
Vogt-Kölln-Str. 30
22527 Hamburg, Germany
haarslev@informatik.uni-hamburg.de

Ralf Möller
Univ. of Appl. Sciences in Wedel
Computer Science Department
Feldstrasse 143
22880 Wedel, Germany
rmoeller@fh-wedel.de

Abstract

We introduce the very expressive description logic $\mathcal{ALCQHI}_{R^+}(\mathcal{D})^-$ providing a limited support for concrete domains. The description logic system Racer supports TBox and ABox reasoning for $\mathcal{ALCQHI}_{R^+}(\mathcal{D})^-$ using a default concrete domain for linear inequations. The adaptation of several important optimization techniques is presented. We conclude the paper with a first proposal for extending $\mathcal{ALCQHI}_{R^+}(\mathcal{D})^-$ by a restricted form of feature chains.

1 Introduction

Reasoning about objects from other domains (so-called concrete domains, e.g. for real numbers) is very important for practical applications, in particular, in the context of the Semantic Web. For instance, one might want to express intervals for integer values ("the price range is between 200 and 300 Euro"), state the relationship between the Fahrenheit and Celsius scales, or describe linear inequalities ("the total price for the three goods must be below 60 Euro"). In [1] the description logic $\mathcal{ALC}(\mathcal{D})$ is investigated and it is shown that, provided a decision procedure for the concrete domain $\mathcal{D}$ exists, the logic $\mathcal{ALC}(\mathcal{D})$ is decidable. Unfortunately, adding concrete domains to expressive description logics (DLs) such as $\mathcal{ALCNH}_{R^+}$ [5] might lead to undecidable inference problems. In [8] it has been shown that $\mathcal{ALCNH}_{R^+}$ extended by a limited form of concrete domains leads to decidable inference problems. This is achieved by disallowing so-called feature chains in $\mathcal{ALCNH}_{R^+}(\mathcal{D})^-$. It is easy to see that the same pragmatic approach can also be applied to very expressive DLs. By analogy to $\mathcal{ALCNH}_{R^+}(\mathcal{D})^-$ the description logic (DL) $\mathcal{ALCQHI}_{R^+}(\mathcal{D})^-$ extends the logic $\mathcal{ALCQHI}_{R^+}$ or $\mathcal{SHIQ}$ [10] with concrete domains.

The DL $\mathcal{ALCQHI}_{R^+}(\mathcal{D})^-$ which is supported by Racer[1] is briefly introduced as follows. We assume five disjoint sets: a set of concept names C, a set of role names R, a set of feature names F, a set of individual names O and a set of names for (concrete) objects O_C. The mutually disjoint subsets P and T of R denote non-transitive and transitive roles, respectively ($R = P \cup T$). $\mathcal{ALCQHI}_{R^+}(\mathcal{D})^-$ is introduced in Figure

[1] Racer download page: http://kogs-www.informatik.uni-hamburg.de/~race/

(a)

Syntax	Semantics
Concepts ($\mathsf{R} \in R$, $\mathsf{S} \in S$, and $\mathsf{f}, \mathsf{f_i} \in F$)	
A	$\mathsf{A}^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$ (A *is a concept name*)
$\neg \mathsf{C}$	$\Delta^{\mathcal{I}} \setminus \mathsf{C}^{\mathcal{I}}$
$\mathsf{C} \sqcap \mathsf{D}$	$\mathsf{C}^{\mathcal{I}} \cap \mathsf{D}^{\mathcal{I}}$
$\mathsf{C} \sqcup \mathsf{D}$	$\mathsf{C}^{\mathcal{I}} \cup \mathsf{D}^{\mathcal{I}}$
$\exists \mathsf{R} . \mathsf{C}$	$\{\mathsf{a} \in \Delta^{\mathcal{I}} \mid \exists \mathsf{b} \in \Delta^{\mathcal{I}} : (\mathsf{a},\mathsf{b}) \in \mathsf{R}^{\mathcal{I}} \wedge \mathsf{b} \in \mathsf{C}^{\mathcal{I}}\}$
$\forall \mathsf{R} . \mathsf{C}$	$\{\mathsf{a} \in \Delta^{\mathcal{I}} \mid \forall \mathsf{b} \in \Delta^{\mathcal{I}} : (\mathsf{a},\mathsf{b}) \in \mathsf{R}^{\mathcal{I}} \Rightarrow \mathsf{b} \in \mathsf{C}^{\mathcal{I}}\}$
$\exists_{\geq n} \mathsf{S} . \mathsf{C}$	$\{\mathsf{a} \in \Delta^{\mathcal{I}} \mid \\|\{\mathsf{y} \mid (\mathsf{x},\mathsf{y}) \in \mathsf{S}^{\mathcal{I}}, \mathsf{y} \in \mathsf{C}^{\mathcal{I}}\}\\| \geq \mathsf{n}\}$
$\exists_{\leq m} \mathsf{S} . \mathsf{C}$	$\{\mathsf{a} \in \Delta^{\mathcal{I}} \mid \\|\{\mathsf{y} \mid (\mathsf{x},\mathsf{y}) \in \mathsf{S}^{\mathcal{I}}, \mathsf{y} \in \mathsf{C}^{\mathcal{I}}\}\\| \leq \mathsf{m}\}$
$\exists \mathsf{f}_1, \ldots, \mathsf{f}_\mathsf{n} . \mathsf{P}$	$\{\mathsf{a} \in \Delta^{\mathcal{I}} \mid \exists \mathsf{x}_1, \ldots, \mathsf{x}_\mathsf{n} \in \Delta^{\mathcal{D}} : (\mathsf{a},\mathsf{x}_1) \in {\mathsf{f}_1}^{\mathcal{I}} \wedge \ldots \wedge (\mathsf{a},\mathsf{x}_\mathsf{n}) \in {\mathsf{f}_\mathsf{n}}^{\mathcal{I}} \wedge (\mathsf{x}_1, \ldots, \mathsf{x}_\mathsf{n}) \in \mathsf{P}^{\mathcal{I}}\}$
$\forall \mathsf{f}_1, \ldots, \mathsf{f}_\mathsf{n} . \mathsf{P}$	$\{\mathsf{a} \in \Delta^{\mathcal{I}} \mid \forall \mathsf{x}_1, \ldots, \mathsf{x}_\mathsf{n} \in \Delta^{\mathcal{D}} : (\mathsf{a},\mathsf{x}_1) \in {\mathsf{f}_1}^{\mathcal{I}} \wedge \ldots \wedge (\mathsf{a},\mathsf{x}_\mathsf{n}) \in {\mathsf{f}_\mathsf{n}}^{\mathcal{I}} \Rightarrow (\mathsf{x}_1, \ldots, \mathsf{x}_\mathsf{n}) \in \mathsf{P}^{\mathcal{I}}\}$
Roles and Features	
R	$\mathsf{R}^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$
f	$\mathsf{f}^{\mathcal{I}} : \Delta^{\mathcal{I}} \to \Delta^{\mathcal{D}}$ (features are partial functions)

$\| \cdot \|$ denotes the cardinality of a set, and $n, m \in \mathbb{N}$ with $n > 1$, $m > 0$.

(b)

Axioms	
Syntax	Satisfied if
$\mathsf{R} \in T$	$\mathsf{R}^{\mathcal{I}} = (\mathsf{R}^{\mathcal{I}})^+$
$\mathsf{R} \sqsubseteq \mathsf{S}$	$\mathsf{R}^{\mathcal{I}} \subseteq \mathsf{S}^{\mathcal{I}}$
$\mathsf{C} \sqsubseteq \mathsf{D}$	$\mathsf{C}^{\mathcal{I}} \subseteq \mathsf{D}^{\mathcal{I}}$

(c)

Assertions ($\mathsf{a}, \mathsf{b} \in O$, $\mathsf{x}, \mathsf{x_i} \in O_C$)	
Syntax	Satisfied if
$\mathsf{a} : \mathsf{C}$	$\mathsf{a}^{\mathcal{I}} \in \mathsf{C}^{\mathcal{I}}$
$(\mathsf{a},\mathsf{b}) : \mathsf{R}$	$(\mathsf{a}^{\mathcal{I}}, \mathsf{b}^{\mathcal{I}}) \in \mathsf{R}^{\mathcal{I}}$
$(\mathsf{a},\mathsf{x}) : \mathsf{f}$	$(\mathsf{a}^{\mathcal{I}}, \alpha(\mathsf{x})) \in \mathsf{f}^{\mathcal{I}}$
$(\mathsf{x}_1, \ldots, \mathsf{x}_\mathsf{n}) : \mathsf{P}$	$(\alpha(\mathsf{x}_1), \ldots, \alpha(\mathsf{x}_\mathsf{n})) \in \mathsf{P}^{\mathcal{I}}$

Figure 1: Syntax and Semantics of $\mathcal{ALCQHI}_{R^+}(\mathcal{D})^-$.

1 using a standard Tarski-style semantics with an interpretation $\mathcal{I}_\mathcal{D} = (\Delta^{\mathcal{I}}, \Delta^{\mathcal{D}}, \cdot^{\mathcal{I}})$ where $\Delta^{\mathcal{I}} \cap \Delta^{\mathcal{D}} = \emptyset$ holds. A variable assignment α maps concrete objects to values in $\Delta^{\mathcal{D}}$.

In accordance with [1] we also define the notion of a concrete domain. 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. The interpretation function maps each predicate name P from $\Phi_\mathcal{D}$ with arity n to a subset $\mathsf{P}^{\mathcal{I}}$ of $\Delta_\mathcal{D}^n$. Concrete objects from O_C are mapped to an element of $\Delta^{\mathcal{D}}$. We assume that $\bot_\mathcal{D}$ is the negation of the predicate $\top_\mathcal{D}$. A concrete domain $\mathcal{D}$ is called *admissible* iff the set of predicate names $\Phi_\mathcal{D}$ is closed under negation and $\Phi_\mathcal{D}$ contains a name $\top_\mathcal{D}$ for $\Delta_\mathcal{D}$, and 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, $\mathsf{P}_\mathsf{i}^{\mathsf{n}_\mathsf{i}} \in \Phi_\mathcal{D}$, $\mathsf{n_i}$ is the arity of $\mathsf{P_i}$, and $\mathsf{x_{jk}}$ is a concrete object).

If $\mathsf{R}, \mathsf{S} \in R$ are role names, then $\mathsf{R} \sqsubseteq \mathsf{S}$ 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 *sub-roles* of a role R. R is called a super-role of S if $\mathsf{S} \in \mathsf{R}^{\downarrow}$. 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 sub-role. Due to undecidability issues number restrictions are only allowed for simple roles (see [10]). In concepts, inverse roles R^{-1} (or S^{-1}) may be used instead of role names R (or S). If C and D are concepts, then $\mathsf{C} \sqsubseteq \mathsf{D}$ is a terminological axiom (*generalized concept inclusion* or *GCI*). A finite set of terminological axioms $\mathcal{T}_{\mathcal{R}}$ is called a *terminology* or *TBox* w.r.t. to a given role hierarchy $\mathcal{R}$.[2] An *ABox* $\mathcal{A}$ is a finite set of assertional axioms as defined in Figure 1c.

An interpretation $\mathcal{I}$ is a *model* of a concept C (or *satisfies* a concept C) iff $\mathsf{C}^{\mathcal{I}} \neq \emptyset$ and for all $\mathsf{R} \in R$ it holds that iff $(x, y) \in \mathsf{R}^{\mathcal{I}}$ then $(y, x) \in (\mathsf{R}^{-1})^{\mathcal{I}}$. An interpretation $\mathcal{I}$ is a model of a TBox $\mathcal{T}$ iff it satisfies all axioms in $\mathcal{T}$ (see Figure 1b). 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 satisfies all assertions in $\mathcal{A}$ (see Figure 1c). Different individuals are mapped to different domain objects (unique name assumption). Note that features are interpreted differently from features in [1]. RACER supports the standard TBox and ABox inference services for $\mathcal{ALCQHI}_{R^+}(\mathcal{D})^-$, which are described in detail in [7].

2 Using Concrete Domains in RACER

RACER's standard concrete domain supports reasoning for linear inequations between rational numbers and interval reasoning (min/max) for integers (the corresponding domain for the range of features, reals or integers, must be declared). The use of concrete domains is illustrated with the following example. For sake of readability and brevity we notate predicates as lambda expressions instead of introducing predicate names. Let age be a feature (of 'type' integer).

$\textbf{teenager} \equiv \mathsf{human} \sqcap \exists\, \mathsf{age} \,.\, \lambda(x)(x \geq 16)$
$\textbf{old\_teenager} \equiv \mathsf{human} \sqcap \exists\, \mathsf{age} \,.\, \lambda(x)(x \geq 18)$

A teenager is a human with an age of at least 16 years and an old teenager is a human with an age of at least 18 years. Asking for the subsumees of $\mathsf{teenager}$ reveals that $\mathsf{old\_teenager}$ is a subsumee of $\mathsf{teenager}$. The next example demonstrates the use of linear inequalities between rational numbers. Let $\mathsf{temp\_celsius}$ and $\mathsf{temp\_fahrenheit}$ be features (of 'type' rational).

$\textbf{human\_with\_fever} \equiv \mathsf{human} \sqcap \exists\, \mathsf{temp\_celsius} \,.\, \lambda(x)(x \geq 38.5)$
$\textbf{seriously\_ill\_human} \equiv \mathsf{human} \sqcap \exists\, \mathsf{temp\_celsius} \,.\, \lambda(x)(x \geq 42.0)$
$\textbf{human\_with\_high\_fever} \equiv \mathsf{human} \sqcap \exists\, \mathsf{temp\_fahrenheit} \,.\, \lambda(x)(x \geq 107.6)$

Obviously, the concept $\mathsf{seriously\_ill\_human}$ is subsumed by $\mathsf{human\_with\_fever}$. However, the relationship between the Celsius and Fahrenheit scales has to be represented. After adding the following global axiom, the intended equivalence between $\mathsf{seriously\_ill\_human}$ and $\mathsf{human\_with\_high\_fever}$ is properly recognized.

$\top \sqsubseteq \exists\, \mathsf{temp\_celsius}, \mathsf{temp\_fahrenheit} \,.\, \lambda(x, y)(y = 1.8 \cdot x + 32)$

Using these TBox axioms the following ABox $\mathcal{A}$ sets up constraints between the individuals eve and doris.

$\{\mathsf{eve} : \mathsf{human}, (\mathsf{eve}, \mathsf{temp\_eve}) : \mathsf{temp\_fahrenheit}, (\mathsf{temp\_eve}) : \lambda(x)(x = 102.56),$
$\mathsf{doris} : \mathsf{human}, (\mathsf{doris}, \mathsf{temp\_doris}) : \mathsf{temp\_celsius}, (\mathsf{temp\_doris}) : \lambda(x)(x = 39.5)\}$

[2]The reference to $\mathcal{R}$ is omitted in the following if we use $\mathcal{T}$.

Now, asking for the direct types of eve and doris reveals that both individuals are instances of human_with_fever. If (temp_eve, temp_doris) : $\lambda(x, y)(x > y)$ is added to $\mathcal{A}$, it causes an inconsistency since eve's temperature is not higher than doris's temperature.

3 Optimizing Concrete Domain Reasoning

Experiments with a prototype implementation [14] of $\mathcal{ALC}(\mathcal{D})$ indicated that concrete domain (CD) reasoning should only be integrated into a DL system supporting at least standard optimization techniques such as dependency-directed backtracking, caching, GCI absorption, and pseudo model merging. In the following we discuss the interaction between CD reasoning and some of these optimization techniques in more detail.

3.1 Incremental State-based Concrete Domain Tester

Initial tests indicated that for real applications, *incremental* constraint satisfaction algorithms have to be explored for dealing with large search spaces. An incremental constraint solver for linear inequations inspired by [11] has been developed and integrated into Racer. This solver is based on a so-called *CD state* keeping track of added CD predicates, their backtracking dependencies, and the internal data structures (vectors, matrices, etc) maintained by the incremental Simplex procedure. The solver is embedded into a recursive ABox tableaux procedure traversing a search tree spawned by ABox assertions. The CD tester is invoked whenever a branch of the search tree is explored which adds a new CD predicate (e.g., see [8] for example tableau rules). In case a clash is discovered, the tableaux procedure has to backtrack to the most recent choice point in the search tree and to explore other alternatives. In order to ensure correctness, it must be possible for the tableaux procedure to restore previous CD states either by undoing the changes or by reverting to a copy of the original state of the choice point. Due to the technical details of the incremental Simplex procedure, it is rather difficult and time consuming to undo the changes to CD states.

Therefore, we decided to keep copies of CD states. The simplest idea is to always save a copy for possible backtracking before a CD state is modified. This works well for small examples but fails for large and complex knowledge bases (KBs) such as Tambis[3]. Another alternative is to save a copy of the current CD state before an alternative at a choice point is explored (e.g., if a disjunct of a disjunction is tested). However, this may result in many unused copies if the search below the choice point does not necessarily adds new CD predicates. The third alternative, which is implemented in Racer, is a refined combination of both approaches. A copy of the current CD state is only saved if a predicate to be added has new backtracking dependencies which are not yet recorded in the current CD state, i.e., a new choice point for backtracking must be observed. Initial tests with the Tambis KB, which has been adapted to concrete domains, suggested that the third strategy significantly reduces the number of copied CD states.

[3]The original Tambis KB uses the $\mathcal{ALCQHI}_{R^+}$ logic, contains ~400 named concepts and over 50 GCIs. For further details see [2]. For our experiments we added concrete domain constructs resulting in the logic $\mathcal{ALCQHI}_{R^+}(\mathcal{D})^-$.

3.2 Dependency-directed Backtracking

The adaptation of dependency-directed backtracking (DDB) to $\mathcal{ALC}(\mathcal{D})$ is described in [15] where the following requirement was identified. In order to allow DDB after a CD clash, the CD tester must identify all minimal, inconsistent sets of concrete predicates (also referred to as clash culprits). These minimal sets define the necessary dependencies for backtracking. If this is not supported by the CD tester or not computationally feasible, DDB must be disabled after a CD clash. In the following we present a relaxed version of this restriction.

Let us assume a state-based CD tester which does not identify all minimal, inconsistent sets. In order to keep the set of recorded predicates and their dependencies as small as possible, the CD tester is used as follows. Before a predicate P is added, its negation $\overline{\mathsf{P}}$ is added to a copy of the current CD state. If $\overline{\mathsf{P}}$ causes a clash, the CD state already entails P. The predicate P is not added to the current CD state and, in particular, the backtracking dependencies of P can be safely ignored. If $\overline{\mathsf{P}}$ does not cause a clash, P is added to the original CD state. In case the CD tester signals an inconsistency, the last predicate added to the CD state is guaranteed to be a clash culprit. The dependencies for backtracking are defined as the union of the dependencies of this culprit and the dependencies recorded for all CD predicates already added to the CD state. With this strategy, a possible overhead is introduced but this is usually compensated by a better backtracking behavior avoiding (thrashing) situations which introduce redundant choice points. Again, initial experiments with the extended Tambis KB indicated an improvement in runtime performance if the improved CD testing is enabled.

3.3 Pseudo Model Merging

Pseudo model merging [9, 8] is known to be a very effective optimization technique for classifying TBoxes. Of course, this technique should also be extended to DLs with concrete domains in order to enable optimized TBox classification.

The adaptation of pseudo model merging to $\mathcal{ALC}(\mathcal{D})$ is reported in [15, 8]. Due to the syntactic restriction for concrete domains in $\mathcal{ALCQHI}_{R^+}(\mathcal{D})^-$, which disallows feature chains, one can extend the algorithms presented in [8] by a kind of "deep CD model merging."

Let $\mathsf{A} \in C$ be a concept name, $\mathsf{R} \in R$ a role name, $\mathsf{F} \in F$ a feature name, and $\mathsf{a} \in O$ an individual name. In order to obtain a *flat pseudo model* for a concept C the consistency of the ABox $\mathcal{A} = \{\mathsf{a} : \mathsf{C}\}$ 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 pseudo model 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.

$$M^{\mathsf{A}} = \{\mathsf{A} \mid \mathsf{a} : \mathsf{A} \in \mathcal{A}'\},\ M^{\neg\mathsf{A}} = \{\mathsf{A} \mid \mathsf{a} : \neg\mathsf{A} \in \mathcal{A}'\},\ M^{\exists} = \{\mathsf{R} \mid\ \mathsf{a} : \exists\,\mathsf{R} \,.\, \mathsf{C} \in \mathcal{A}'\},$$
$$M^{\forall} = \{\mathsf{R} \mid \mathsf{a} : \forall\,\mathsf{R} \,.\, \mathsf{C} \in \mathcal{A}'\},\ M^{\forall\mathsf{f}} = \{\forall\,\mathsf{f}_1, \ldots, \mathsf{f}_\mathsf{n} \,.\, \mathsf{P} \mid \mathsf{a} : \forall\,\mathsf{f}_1, \ldots, \mathsf{f}_\mathsf{n} \,.\, \mathsf{P} \in \mathcal{A}'\},$$
$$M^{\exists\mathsf{f}} = \{\exists\,\mathsf{f}_1, \ldots, \mathsf{f}_\mathsf{n} \,.\, \mathsf{P} \mid \mathsf{a} : \exists\,\mathsf{f}_1, \ldots, \mathsf{f}_\mathsf{n} \,.\, \mathsf{P} \in \mathcal{A}'\}$$

The procedure **$\mathcal{ALC}(\mathcal{D})$-mergable** (see Procedure 1) implements the new flat model merging test for $\mathcal{ALC}(\mathcal{D})$ for a given non-empty set of pseudo models MS.

Procedure 1 $\mathcal{ALC}(\mathcal{D})$**-mergable(**$MS$**)**

if $\bot \in MS$ **then**
  return *false*
for all pairs $\{M_1, M_2\} \subseteq MS$ **do**
  if $(M_1^{\mathsf{A}} \cap M_2^{\neg\mathsf{A}}) \neq \emptyset \vee (M_1^{\neg\mathsf{A}} \cap M_2^{\mathsf{A}}) \neq \emptyset \vee (M_1^{\exists} \cap M_2^{\forall}) \neq \emptyset \vee (M_1^{\forall} \cap M_2^{\exists}) \neq \emptyset$ **then**
    return *false*
cd_state ← create_empty_cd_state()
for all $M \in MS$ **do**
  for all $\mathsf{C} \in M^{\exists \mathsf{f}}$ **do**
    ⟨sat,new_cd_state⟩ ← predicates_satisfiable(C,cd_state)
    if ¬ sat **then**
      return *false*
    cd_state ← new_cd_state
for all $M \in MS$ **do**
  for all $\mathsf{C} \in M^{\forall \mathsf{f}}$ **do**
    if all_applicable(C,cd_state) **then**
      ⟨sat,new_cd_state⟩ ← predicates_satisfiable(C,cd_state)
      if ¬ sat **then**
        return *false*
      cd_state ← new_cd_state
return *true*

$\mathcal{ALC}(\mathcal{D})$**-mergable** uses the procedure **predicates_satisfiable** which decides the CD satisfiability. The parameter C must be a concept of the form $\forall\, \mathsf{f}_1, \ldots, \mathsf{f}_n \,.\, \mathsf{P}$ or $\exists\, \mathsf{f}_1, \ldots, \mathsf{f}_n \,.\, \mathsf{P}$. The procedure **all_applicable** decides whether a concept of the form $\forall\, \mathsf{f}_1, \ldots, \mathsf{f}_n \,.\, \mathsf{P}$ is "applicable" to a CD state. A proof for the soundness of $\mathcal{ALC}(\mathcal{D})$**-mergable** can be easily adapted from the one given in [8] if we assume that **predicates_satisfiable** decides the satisfiability of a finite conjunction of CD predicates representing linear inequations.

3.4 Absorption of Domain and Range Restrictions

The absorption of global TBox axioms representing so-called *domain* or *range* restrictions of roles [6] can be extended to concrete features accordingly. For instance, Racer absorbs "feature domain" axioms of the form $\exists\, \mathsf{f} \,.\, \top_{\mathcal{D}} \sqsubseteq \mathsf{C}$ and "feature range" axioms of the form $\top \sqsubseteq \forall\, \mathsf{f} \,.\, \mathsf{P}$. For instance, this absorption technique demonstrated a runtime improvement of at least one order of magnitude for classifying a modified version of the "Stereo KB" developed for the Classic system. This KB[4] describes knowledge about the configuration of HiFi stereo systems and our modified version has been adapted to concrete domains.

3.5 Adding a Restricted Form of Feature Chains

The last subsection of this paper discusses a proposal to relax the syntax restriction of $\mathcal{ALCQHI}_{R^+}(\mathcal{D})^-$ w.r.t. feature chains. From a modeling point of view the restriction to globally disallow feature chains is rather severe. For instance, this restriction prevents the proper representation of "nested data structures" such as vectors where a

[4]The stereo KB contains over 700 named concepts and over 50 GCIs. For more details see [12].

vector is described by two two-dimensional points (called tip and toe) which, in turn, are described by two coordinates (called x and y), i.e., one would like to use a concept $\exists\,(\mathsf{tip}\cdot\mathsf{x}),(\mathsf{toe}\cdot\mathsf{x})\,.\,\lambda(x,y)(x=y)$ describing a vertical vector (tip, toe, x, y are feature names). Currently, these "data structures" have to be flattened in order to avoid the need for feature chains. However, the "flattening" solution is rather unsatisfactory and results in less flexible descriptions. For instance, if a quadrangle should be specified by 4 vectors whose end points do pairwise coincide, the vectors have to be flattened into 16 different features describing the x and y coordinates of the tips and toes of the 4 vectors. In the flattened representation the notion of a vector is no longer present. Thus, it is not possible to express role value restrictions for vectors.

In the following we propose for $\mathcal{ALCQHI}_{R^+}(\mathcal{D})^-$ a relaxed restriction w.r.t. to feature chains. The relaxation is motivated by the observation that one has to avoid concept descriptions using feature chains, which enforce infinite models. This is illustrated by the following axioms describing ordered lists of rationals ($\mathsf{has\_val}$, $\mathsf{has\_succ}$ are always feature names).

$\textsf{\textbf{ordered\_list}} \sqsubseteq \ \exists\,\mathsf{has\_val},\mathsf{has\_succ}\cdot\mathsf{has\_val}\,.\,\lambda(x,y)(x+1=y)$

$\top \sqsubseteq \ \forall\,\mathsf{has\_succ}\,.\,\mathsf{ordered\_list}$

A second variant is an axiom introducing a terminological cycle for $\mathsf{ordered\_list}$.

$\textsf{\textbf{ordered\_list}} \sqsubseteq$

$\exists\,(\mathsf{has\_val}),(\mathsf{has\_succ}\cdot\mathsf{has\_val})\,.\,\lambda(x,y)(x+1=y) \sqcap \exists\,\mathsf{has\_succ}\,.\,\mathsf{ordered\_list}$

A third variant uses a transitive super-role $\mathsf{precedes}$ of $\mathsf{has\_succ}$.

$\textsf{\textbf{ordered\_list}} \sqsubseteq$

$\exists\,(\mathsf{has\_val}),(\mathsf{has\_succ}\cdot\mathsf{has\_val})\,.\,\lambda(x,y)(x+1=y) \sqcap$

$\forall\,\mathsf{precedes}\,.\,\exists\,(\mathsf{has\_val}),(\mathsf{has\_succ}\cdot\mathsf{has\_val})\,.\,\lambda(\mathsf{x},\mathsf{y})(\mathsf{x}+1=\mathsf{y})$

However, feature chains are usually not "harmful" in concepts with finite models. This observation motivated the following relaxed restriction. We assume an absorption process for TBox axioms which partitions a TBox $\mathcal{T}$ into two sets. One TBox partition ($\mathcal{T}_u$) contains only axioms which can be dealt with by the lazy unfolding technique. The other TBox ($\mathcal{T}_m$) contains the remaining axioms. The TBox $\mathcal{T}$ is called admissible if the following condition is met for both partitions. Let u_1,...,u_n be feature chains. Feature chains are only allowed in concepts of the form $\exists\,\mathsf{u}_1,\ldots,\mathsf{u}_\mathsf{n}\,.\,\mathsf{P}$ if it holds that no feature mentioned in the chains u_i (1) has a super-role and (2) is involved in a terminological cycle. Additionally, the conditions (1-2) must also hold for all concept terms of the form $\forall\,\mathsf{u}_1,\ldots,\mathsf{u}_\mathsf{n}\,.\,\mathsf{P}$ occurring in the TBox partition $\mathcal{T}_u$.

4 Outlook

The proposed techniques are implemented in Racer and are currently empirically evaluated in more detail. The proposed relaxation of the use of feature chains is subject to ongoing work.

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] S. Bechhofer, Ian Horrocks, C. Goble, and R. Stevens. OilEd: A reason-able ontology editor for the semantic web. In T. Eiter F. Baader, G. Brewka, editor, *Proceedings of KI 2001: Advances in Artificial Intelligence Joint German/Austrian Conference on AI*, volume 2174 of *LNAI*, page 396 ff., 2001.

[3] 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*, April 2000.

[4] R. Goré, A. Leitsch, and T. Nipkow, editors. *Proceedings of the International Joint Conference on Automated Reasoning, IJCAR'2001, June 18-23, 2001, Siena, Italy*, Lecture Notes in Computer Science. Springer-Verlag, June 2001.

[5] V. Haarslev and R. Möller. Expressive ABox reasoning with number restrictions, role hierarchies, and transitively closed roles. In Cohn et al. [3], pages 273–284.

[6] V. Haarslev and R. Möller. High performance reasoning with very large knowledge bases: A practical case study. In Nebel [13], pages 161–166.

[7] V. Haarslev and R. Möller. RACER user's guide and reference manual version 1.6. Technical report, University of Hamburg, Computer Science Department, July 2001.

[8] V. Haarslev, R. Möller, and A.-Y. Turhan. Exploiting pseudo models for TBox and ABox reasoning in expressive description logics. In Goré et al. [4], pages 61–75.

[9] I. Horrocks. *Optimising Tableaux Decision Procedures for Description Logics.* PhD thesis, University of Manchester, 1997.

[10] I. Horrocks, U. Sattler, and S. Tobies. Reasoning with individuals for the description logic $\mathcal{SHIQ}$. In David MacAllester, editor, *Proceedings of the 17th International Conference on Automated Deduction (CADE-17)*, Lecture Notes in Computer Science, Germany, 2000. Springer Verlag.

[11] J. Jaffar, S. Michaylov, P.J. Stuckey, and R.H.C. Yap. The CLP($\Re$) language and system. *ACM Transactions on Programming Languages and Systems*, 14(3):339–395, July 1992.

[12] D.L. McGuinness, L.A. Resnick, and C. Isbell. Description logic in practice: A CLASSIC application. In *IJCAI'95, 14th International Conference on Artificial Intelligence*, pages 2045–2046. Morgan-Kaufmann Publ., 1995.

[13] B. Nebel, editor. *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence, IJCAI-01, August 4-10, 2001, Seattle, Washington, USA*, June 2001.

[14] A.-Y. Turhan. Design and implementation of description logic provers for $\mathcal{ALC}(\mathcal{D})$ and $\mathcal{ALCRP}(\mathcal{D})$ (in german), October 1998. Bachelors Thesis (Studienarbeit).

[15] A.-Y. Turhan and V. Haarslev. Adapting optimization techniques to description logics with concrete domains. In F. Baader and U. Sattler, editors, *Proceedings of the International Workshop on Description Logics (DL'2000), August 17 - August 19, 2000, Aachen, Germany*, pages 247–256, August 2000.