1

Description Logics with Concrete Domains—A Survey

CARSTEN LUTZ

ABSTRACT. Description logics (DLs) are a family of logical formalisms that have initially been designed for the representation of conceptual knowledge in artificial intelligence and are closely related to modal logics. In the last two decades, DLs have been successfully applied in a wide range of interesting application areas. In most of these applications, it is important to equip DLs with expressive means that allow to describe "concrete qualities" of real-world objects such as their weight, temperature, and spatial extension. The standard approach is to augment description logics with so-called concrete domains, which consist of a set (say, the rational numbers), and a set of n-ary predicates with a fixed extension over this set. The "interface" between the DL and the concrete domain is then provided by a new logical constructor that has, to the best of our knowledge, no counterpart in modal logics. In this paper, we give an overview over description logics with concrete domains and summarize decidability and complexity results from the literature.

1 Introduction

Description logics (DLs) are a family of logical formalisms that originated in the field of artificial intelligence as a tool for the representation of conceptual knowledge. Since then, DLs have been successfully used in a wide range of application areas such as knowledge representation, reasoning about class-based formalisms (e.g. conceptual database models and UML diagrams), and ontology engineering in the context of the semantic web [Baader 1999; Calvanese *et al.* 1998; Baader *et al.* 2002a]. The basic syntactic entity of description logics are *concepts*, which are constructed from concept names (unary predicates) and role names (binary relations) using the set of concept and role constructors provided by a particular DL. For example, the following concept is formulated in the basic propositionally closed description logic $\mathcal{ALC}$ and could be used, e.g., in a knowledge-based process engineering application (as in [Sattler 1998; Molitor 2000]) to describe a production process that has an expensive (specially trained) operator:

$$\mathsf{Process} \sqcap \forall \mathsf{subproc}.\mathsf{Process} \sqcap \exists \mathsf{operator}.(\mathsf{Human} \sqcap \mathsf{Expensive})$$

Advances in Modal Logic, Volume 4, 1–??.

In this example, Process, Human, and Expensive are concept names while subproc and operator are role names.

Viewed from a logical perspective, description logics are closely related to modal logics [Schild 1991; Giacomo & Lenzerini 1994]. For example, the DL $\mathcal{ALC}$ can be viewed as a notational variant of the modal logic K_ω, i.e., multimodal K with infinitely many accessibility relations: concept names correspond to propositional variables, role names correspond to (names for) accessibility relations, the $\forall$ constructor of $\mathcal{ALC}$ can be read as a modal box operator, and $\exists$ can be read as a diamond. However, there also exist several means of expressivity that are frequently used in description logics, but usually not considered in modal logics.

An important example are so-called *concrete domains*, which allow the integration of "concrete qualities" such as numbers, time intervals, and strings into description logic concepts. Suppose, for example, that we want to refine the description of a process given above by replacing the concept name Expensive with a concept expressing that the process operator earns at least 20 euro per hour. Then we need a proper way to talk about numbers such as "20" and comparisons between numbers such as "at least 20 euro". As another example, we may want to express that the time interval describing the working time of the operator should contain the time interval describing the execution time of the process. Here we obviously need to represent time intervals and relations between them.

The need for extending the expressive power of DLs in the described direction arises in almost all relevant application areas, let us review two (more) examples:

1. Semantic web. In this application, DLs are used to describe the contents of web pages in order to facilitate the development of more powerful web services such as advanced search engines [Baader *et al.* 2002a; Berners-Lee *et al.* 2001]. It is obvious and has always been emphasized that the representation of "concrete datatypes" such as numbers and strings is an important issue [Fensel *et al.* 2000; Horrocks & Patel-Schneider 2001]: if, for example, we want to describe the web page of a wine seller, then we need numbers to represent vintages and prices, and strings to represent the names of regions and wine producers. It should be clear that such concrete datatypes are precisely what we have described as "concrete qualities".

2. Conceptual database models. Entity Relationship (ER) diagrams are the predominant formalism for constructing conceptual models of relational databases [Chen 1976; Teorey 1990]. For example, an ER diagram could describe two entities Employee and Company related by a relationship employs such that each Employee is employed by exactly one Company, and each

Company employs at least one Employee. DLs can be used to encode and reason about ER diagrams, which allows to detect inconsistencies and implications that are only implicitly represented in the diagram [Calvanese *et al.* 1998; Franconi & Ng 2000]. However, the standard translation of ER diagrams into DLs does not take into account so-called "numerical attribute dependencies", which can e.g. be used to express that no Employee was hired prior to his Company's founding. As argued in [Lutz 2002e], it is important to include these dependencies when using DLs for reasoning about ER diagrams since they can be an (additional) source for inconsistencies and unnoticed ramifications. In order to do this, the target DL must be able to represent "concrete" objects such as numbers and comparisons between numbers.

The necessity of representing concrete qualities in description logics has been realized almost since the beginnings of the field, and, indeed, many early description logic reasoners such as MESON [Edelmann & Owsnicki 1986] and CLASSIC [Brachman *et al.* 1991] provided for "ad hoc" solutions of this problem. The first formal treatment of the issue was presented by Baader and Hanschke in [1991], who proposed to extend the description logic $\mathcal{ALC}$ with concrete domains. Formally, a concrete domain consists of a set such as the natural numbers and a set of predicates such as the binary "$<$" and the ternary "$+$" with a *fixed* extension over this set. Enriching $\mathcal{ALC}$ with such a concrete domain $\mathcal{D}$, we obtain the basic DL with concrete domains $\mathcal{ALC}(\mathcal{D})$. More precisely, $\mathcal{ALC}(\mathcal{D})$ is obtained from $\mathcal{ALC}$ by augmenting it with

- *abstract features*, i.e. roles interpreted as functional relations;
- *concrete features*: a new syntactic type that is interpreted as a partial function from the logical domain into the concrete domain;
- a new concept constructor that allows to describe constraints on concrete values using predicates from the concrete domain.

Let us view two example $\mathcal{ALC}(\mathcal{D})$-concepts: the concept

$$\mathsf{Process} \sqcap \forall \mathsf{subproc}.\mathsf{Process} \sqcap \exists \mathsf{operator}.(\mathsf{Human} \sqcap \exists \mathsf{wage}.\geq_{20})$$

refines the process description from above by replacing the concept name Expensive with a concrete domain-based description of the operator's wage—which is at least 20 euro per hour. In this example, operator is an abstract feature while wage is a concrete feature. We use a concrete domain based on the natural numbers and assume that $\geq_{20}$ is a unary predicate with the obvious extension. The (sub)concept $\exists \mathsf{wage}.\geq_{20}$ is an instantiation of the concrete domain constructor and must not be confused with the existential restriction as used in $\exists$operator.Human. Observe that concrete features such as wage are the "link" between the description logic and the concrete domain: they allow to associate concrete values such as numbers with logical objects such as the one representing the operator in the above example.

In the second concept, we use a concrete domain based on time intervals to describe a constraint on the execution time of processes as proposed above:

$$\mathsf{Process} \sqcap \forall \mathsf{subproc}.\mathsf{Process} \sqcap \exists(\mathsf{operator}\ \mathsf{worktime}), (\mathsf{exectime}).\mathsf{contains}$$

Here, worktime and exectime (for "execution time") are concrete features, and contains is a binary concrete domain predicate. The last conjunct is an instantiation of the concrete domain constructor expressing that the time interval describing the working time of the operator contains the time interval describing the execution time of the process.

Since their first appearance in 1991, description logics with concrete domains have been extensively studied. The purpose of this paper is to survey the proposed logics and available results, focusing on decidability and computational complexity. It is organized as follows: in Section 2, we formally introduce concrete domains and the description logic $\mathcal{ALC}(\mathcal{D})$. Section 3 discusses results that have been obtained for $\mathcal{ALC}(\mathcal{D})$ and several of its extensions: in Section 3.1, we treat $\mathcal{ALC}(\mathcal{D})$ itself, Section 3.2 is concerned with extensions considered "standard" in the area of description logics, and Section 3.3 focuses on specifically concrete-domain related extensions. Most of the discussed results do not consider a particular concrete domain, but are of a general nature. Finally, Section 4 gives a brief overview over concrete domains that have been proposed in the literature.

2 The Description Logic $\mathcal{ALC}(\mathcal{D})$

In this section, we formally introduce Baader and Hanschke's basic description logic with concrete domains $\mathcal{ALC}(\mathcal{D})$ [1991]. To do this, we must first define the underlying notion of concrete domains.

DEFINITION 1 (Concrete Domain) *A* concrete domain $\mathcal{D}$ *is a pair* $(\Delta_\mathcal{D}, \Phi_\mathcal{D})$, *where* $\Delta_\mathcal{D}$ *is a set and* $\Phi_\mathcal{D}$ *a set of predicate names. Each predicate name* $P \in \Phi_\mathcal{D}$ *is associated with an arity* n *and an* n*-ary predicate* $P^\mathcal{D} \subseteq \Delta^n_\mathcal{D}$.

For many application areas, the most interesting concrete domains are *numerical* ones. Hence, let us introduce a typical numerical concrete domain Q to illustrate Definition 1: as the set Δ_Q, we use the rational numbers $\mathbb{Q}$. The following predicates are available:

– unary predicates P_q for each $P \in \{<, \leq, =, \neq, \geq, >\}$ and each $q \in \mathbb{Q}$ with $(P_q)^{\mathsf{Q}} = \{q' \in \mathbb{Q} \mid q' \; P \; q\}$;

– binary predicates $<, \leq, =, \neq, \geq, >$ with the obvious extension;

– ternary predicates $+$ and $\overline{+}$ with $(+)^{\mathsf{Q}} = \{(q, q', q'') \in \mathbb{Q}^3 \mid q+q' = q''\}$ and $(\overline{+})^{\mathsf{Q}} = \mathbb{Q}^3 \setminus (+)^{\mathsf{Q}}$;

– a unary predicate $\top_{\mathsf{Q}}$ with $(\top_{\mathsf{Q}})^{\mathsf{Q}} = \mathbb{Q}$ and a unary predicate $\bot_{\mathsf{Q}}$ with $(\bot_{\mathsf{Q}})^{\mathsf{Q}} = \emptyset$.

The presence of the predicates $\top_{\mathsf{Q}}$ and $\bot_{\mathsf{Q}}$ and of the negation of the "+" predicate is related to the *admissibility* of concrete domains and will be discussed in Section 3.1. We will further discuss the concrete domain Q and its relatives in Section 4.

DEFINITION 2 ($\mathcal{ALC}(\mathcal{D})$ Syntax) *Let* $\mathsf{N_C}$, $\mathsf{N_R}$, *and* $\mathsf{N_{cF}}$ *be pairwise disjoint and countably infinite sets of* concept names, role names, *and* concrete features. *Furthermore, let* $\mathsf{N_{aF}}$ *be a countably infinite subset of* $\mathsf{N_R}$. *The elements of* $\mathsf{N_{aF}}$ *are called* abstract features. *A* path *u is a composition $f_1 \cdots f_n g$ of n abstract features $f_1, \ldots, f_n$ ($n \geq 0$) and a concrete feature g. For $\mathcal{D}$ a concrete domain, the set of $\mathcal{ALC}(\mathcal{D})$-concepts is the smallest set such that*

– *every concept name is a concept, and*
– *if C and D are concepts, R is a role name, g is a concrete feature, $u_1, \ldots, u_n$ are paths, and $P \in \Phi_{\mathcal{D}}$ is a predicate of arity n, then the following expressions are also concepts: $\neg C$, $C \sqcap D$,$C \sqcup D$, $\exists R.C$, $\forall R.C$, $\exists u_1, \ldots, u_n.P$, and $g\uparrow$.*

As usual, we use $\top$ as abbreviation for an arbitrary propositional tautology and $\bot$ as abbreviation for $\neg\top$. Additionally, if $u = f_1 \cdots f_k g$ is a path then $u\uparrow$ is used as abbreviation for $\forall f_1. \cdots .\forall f_k.g\uparrow$. As an example $\mathcal{ALC}(\mathsf{Q})$-concept, consider the process description

$$\begin{aligned}&\mathsf{Process} \sqcap \forall \mathsf{subproc}.\mathsf{Process} \sqcap \exists \mathsf{operator}.(\mathsf{Human} \sqcap \exists \mathsf{wage}.\geq_{20})\\ &\sqcap \exists (\mathsf{operator}\ \mathsf{wage}), (\mathsf{cost}).\leq,\end{aligned}$$

where the second line states that the hourly cost of the process is at least as high as the hourly wage of its operator. We now introduce the semantics of $\mathcal{ALC}(\mathcal{D})$-concepts and the relevant reasoning problems.

DEFINITION 3 ($\mathcal{ALC}(\mathcal{D})$ Semantics)*An* interpretation *$\mathcal{I}$ is a pair $(\Delta_{\mathcal{I}}, \cdot^{\mathcal{I}})$, where $\Delta_{\mathcal{I}}$ is a set called the* domain *and $\cdot^{\mathcal{I}}$ is the* interpretation function. *The interpretation function maps*

– *each concept name* C *to a subset* $C^{\mathcal{I}}$ *of* $\Delta_{\mathcal{I}}$,
– *each role name* R *to a subset* $R^{\mathcal{I}}$ *of* $\Delta_{\mathcal{I}} \times \Delta_{\mathcal{I}}$,
– *each abstract feature* f *to a partial function* $f^{\mathcal{I}}$ *from* $\Delta_{\mathcal{I}}$ *to* $\Delta_{\mathcal{I}}$*, and*
– *each concrete feature* g *to a partial function* $g^{\mathcal{I}}$ *from* $\Delta_{\mathcal{I}}$ *to* $\Delta_{\mathcal{D}}$.

If $u = f_1 \cdots f_n g$ *is a path, then* $u^{\mathcal{I}}(d)$ *is defined as* $g^{\mathcal{I}}(f_n^{\mathcal{I}} \cdots (f_1^{\mathcal{I}}(d)) \cdots)$. *The interpretation function is extended to arbitrary concepts as follows:*

$$\begin{aligned}
(\neg C)^{\mathcal{I}} &:= \Delta_{\mathcal{I}} \setminus C^{\mathcal{I}} \\
(C \sqcap D)^{\mathcal{I}} &:= C^{\mathcal{I}} \cap D^{\mathcal{I}} \\
(C \sqcup D)^{\mathcal{I}} &:= C^{\mathcal{I}} \cup D^{\mathcal{I}} \\
(\exists R.C)^{\mathcal{I}} &:= \{d \in \Delta_{\mathcal{I}} \mid \{e \mid (d,e) \in R^{\mathcal{I}}\} \cap C^{\mathcal{I}} \neq \emptyset\} \\
(\forall R.C)^{\mathcal{I}} &:= \{d \in \Delta_{\mathcal{I}} \mid \{e \mid (d,e) \in R^{\mathcal{I}}\} \subseteq C^{\mathcal{I}}\} \\
(\exists u_1, \ldots, u_n.P)^{\mathcal{I}} &:= \{d \in \Delta_{\mathcal{I}} \mid \exists x_1, \ldots, x_n \in \Delta_{\mathcal{D}} : u_i^{\mathcal{I}}(d) = x_i \text{ for } 1 \leq i \leq n \\
&\qquad\qquad \text{and } (x_1, \ldots, x_n) \in P^{\mathcal{D}}\} \\
(g\uparrow)^{\mathcal{I}} &:= \{d \in \Delta_{\mathcal{I}} \mid g^{\mathcal{I}}(d) \text{ undefined}\}
\end{aligned}$$

Let $\mathcal{I}$ *be an interpretation. Then* $\mathcal{I}$ *is a* model *of a concept* C *iff* $C^{\mathcal{I}} \neq \emptyset$. *A concept* C *is* satisfiable *iff* C *has a model. A concept* C *is* subsumed by *a concept* D *(written* $C \sqsubseteq D$*) iff* $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ *for all interpretations* $\mathcal{I}$.

While satisfiability is familiar from modal and classical logics, subsumption deserves a brief comment: this reasoning task is rather important in description logics since DLs are frequently used to capture the terminological knowledge of an application domain, and subsumption can then be used to arrange the defined notions (represented by concepts) in a taxonomy. Logically, subsumption can obviously be understood as the validity of implications. It should thus be clear that, in $\mathcal{ALC}(\mathcal{D})$, concept subsumption can be reduced to concept (un)satisfiability and vice versa: $C \sqsubseteq D$ iff $C \sqcap \neg D$ is unsatisfiable and C is satisfiable iff $C \not\sqsubseteq \bot$.

It is not hard to see that "the $\mathcal{ALC}$ part" of $\mathcal{ALC}(\mathcal{D})$ is a syntactical variant of multimodal K (see Section 1). However, to the best of our knowledge, the concrete domain constructor has no counterpart in modal logic. Moreover, even for very simple concrete domains $\mathcal{D}$ there does not exist a translation from $\mathcal{ALC}(\mathcal{D})$-concepts into formulas of the two-variable fragment of first-order logic or of the guarded fragment—a property enjoyed by most modal and description logics [van Benthem 1983; Borgida 1996]. The reason for this is that we admit paths of length greater one inside the concrete domain constructor.

For most application areas, the reasoning tasks "concept satisfiability" and "subsumption" have to take into account so-called TBoxes. Such TBoxes are sets of concept equations, which are used to store terminological knowledge and background knowledge about the application domain. For example, we could use a TBox to define the notion "expensive process" by writing

$$\mathsf{ExpensiveProcess} \doteq \mathsf{Process} \sqcap \exists \mathsf{cost}.\geq_{20}$$

Moreover, we could capture the "background knowledge" that every process controlled by an expensive operator is an expensive process:

$$\top \doteq (\mathsf{Process} \sqcap \exists \mathsf{operator}.\exists \mathsf{wage}.\geq_{20}) \rightarrow \mathsf{ExpensiveProcess}$$

In the DL literature, there exist various TBox formalisms with vast differences in expressive power. In this paper, we will only consider the two TBox formalisms that are used most frequently.

DEFINITION 4 (TBox) *A* concept equation *is an expression* $C \doteq D$*, where* C *and* D *are concepts. A* general TBox *is a finite set of concept equations.*

A concept equation $C \doteq D$ *is called a* concept definition *if* C *is a concept name. A finite set of concept definitions* $\mathcal{T}$ *is called an* acyclic TBox *if the following conditions are satisfied:*

1. *the left-hand sides of concept definitions are unique, i.e., if* $\{A \doteq C,\ A' \doteq C'\} \subseteq \mathcal{T}$*, then* $C \neq C'$ *implies* $A \neq A'$*;*

2. $\mathcal{T}$ *is acyclic: there are no concept definitions* $\{A_0 \doteq C_0,\ \ldots,\ A_{k-1} \doteq C_{k-1}\} \subseteq \mathcal{T}$ *such that the concept name* A_i *occurs in* $C_{i+1 \bmod k}$ *for* $i < k$*.*

An interpretation $\mathcal{I}$ *is a* model *of a (general or acyclic) TBox* $\mathcal{T}$ *if* $C^{\mathcal{I}} = D^{\mathcal{I}}$ *for all* $C \doteq D \in \mathcal{T}$*. A concept* C *is* satisfiable w.r.t. a TBox $\mathcal{T}$ *iff* C *and* $\mathcal{T}$ *have a common model. A concept* C *is* subsumed by *a concept* D w.r.t. a TBox $\mathcal{T}$ *(written* $C \sqsubseteq_{\mathcal{T}} D$*) iff* $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ *for all models* $\mathcal{I}$ *of* $\mathcal{T}$*.*

From a modal logic perspective, the expressive power provided by general TBoxes is closely related to the expressiveness of the universal modality—see e.g. Section 2.2.1 of [Lutz 2002b] for a thorough discussion. While general TBoxes are a rather powerful tool, the expressive power provided by acyclic TBoxes is relatively weak: due to acyclicity, they can be viewed as macro definitions, i.e., as providing a set of abbreviations for concepts. As we will see in Section 3.2, acyclic TBoxes can also be expanded like macros. Note, however, that acyclic TBoxes are still powerful enough to define terminologies as in the first example presented above.

To distinguish concept satisfiability without TBoxes from concept satisfiability with TBoxes, we will in the following sometimes use the term "pure concept satisfiability" to refer to the former.

3 Description Logics with Concrete Domains

In this section, we consider the basic description logic with concrete domains $\mathcal{ALC}(\mathcal{D})$ and several of its extensions. We start with $\mathcal{ALC}(\mathcal{D})$ itself and then discuss "standard extensions" that are frequently considered in description logics and, in principle, independent of concrete domains. Finally, we give an overview over extensions of $\mathcal{ALC}(\mathcal{D})$ that concern the "concrete domain part" of this logic.

3.1 The Basic Formalism

In their original 1991 paper, Baader and Hanschke present a tableau algorithm that is capable of deciding (pure) $\mathcal{ALC}(\mathcal{D})$-concept satisfiability. Using the reduction from the previous section, this algorithm also yields a decision procedure for concept subsumption. Baader and Hanschke's decidability result is a rather general one since it does not concern a particular concrete domain, but applies to any concrete domain that satisfies some weak conditions. These conditions are derived from the fact that any satisfiability algorithm *not* committing itself to a particular concrete domain must call some concrete domain reasoner as a subprocedure via a well-defined "interface". In the case of Baader and Hanschke's tableau algorithm, such a concrete domain reasoner is required to decide the satisfiability of finite conjunctions of concrete domain predicates. This leads to the notion of *admissibility.*

DEFINITION 5 (Admissible) *Let $\mathcal{D}$ be a concrete domain and V a set of variables. A $\mathcal{D}$-*conjunction *is a predicate conjunction of the form*

$$c = \bigwedge_{i<k} (x_0^{(i)}, \ldots, x_{n_i}^{(i)}) : P_i,$$

where P_i is an n_i-ary predicate for $i < k$ and the $x_j^{(i)}$ are variables from V. A $\mathcal{D}$-conjunction c is satisfiable *iff there exists a function δ mapping the variables in c to elements of $\Delta_\mathcal{D}$ such that $(\delta(x_0^{(i)}), \ldots, \delta(x_{n_i}^{(i)})) \in P_i^\mathcal{D}$ for each $i < k$. Such a function is called a* solution *for c. We say that the concrete domain $\mathcal{D}$ is* admissible *iff*

1. *its set of predicate names is closed under negation and contains a name $\top_\mathcal{D}$ for $\Delta_\mathcal{D}$ and*

2. *the satisfiability of $\mathcal{D}$-conjunctions is decidable.*

We refer to the satisfiability of $\mathcal{D}$-conjunctions as $\mathcal{D}$-satisfiability.

Property 1 of admissibility has to be satisfied since the description logic $\mathcal{ALC}(\mathcal{D})$ provides for negation. For example, the concept

$$\neg(g_1\uparrow) \sqcap \neg(g_2\uparrow) \sqcap \neg(\exists g_1, g_2.<)$$

expresses that $g_1^{\mathcal{I}} \geq g_2^{\mathcal{I}}$ without explicitly using a "$\geq$" predicate, and such information must be passed to the concrete domain reasoner. Note that the concrete domain Q presented in Section 2 satisfies Property 1 of admissibility—in Section 4, we will see that Property 2 is also satisfied. The result obtained in [Baader & Hanschke 1991] can now be formulated as follows:

THEOREM 6 (Baader, Hanschke) *Pure $\mathcal{ALC}(\mathcal{D})$-concept satisfiability and subsumption are decidable if $\mathcal{D}$ is admissible.*

We should briefly comment on a minor difference between the logic $\mathcal{ALC}(\mathcal{D})$ as defined in [Baader & Hanschke 1991] and in Section 2: Baader and Hanschke's variant of $\mathcal{ALC}(\mathcal{D})$ uses only a single type of feature that is interpreted in partial functions from $\Delta_{\mathcal{I}}$ to $\Delta_{\mathcal{I}} \times \Delta_{\mathcal{D}}$ and thus combines our abstract and concrete features. It is not very hard to see that the difference in expressivity is negligible. However, the separation of abstract and concrete features necessitates the presence of the $g\uparrow$ constructor: without this constructor, we would not be able to remove negations in front of the concrete domain constructor when converting $\mathcal{ALC}(\mathcal{D})$-concepts into equivalent ones in negation normal form (NNF), for details see Section 3.3.*

The complexity of reasoning with $\mathcal{ALC}(\mathcal{D})$ has been analyzed in [Lutz 2002d]. There, the tableau algorithm of Baader and Hanschke is refined by using the so-called *tracing technique*: instead of keeping entire tableaux in memory (which may become exponentially large), a tree-shaped tableau is constructed in a depth-first manner keeping only paths of the tree in memory. Since such paths are of at most polynomial length, this allows to devise a PSPACE algorithm. However, the complexity of reasoning with $\mathcal{ALC}(\mathcal{D})$ clearly depends on the complexity of $\mathcal{D}$-satisfiability:

THEOREM 7 *Pure $\mathcal{ALC}(\mathcal{D})$-concept satisfiability and subsumption are* PSPACE*-complete if $\mathcal{D}$ is admissible and $\mathcal{D}$-satisfiability is in* PSPACE.

*A concept is in NNF if negation does only occur in front of concept names. This normal form is frequently used to devise decision procedures for DLs.

Thus, reasoning with $\mathcal{ALC}(\mathcal{D})$ is not harder than reasoning with $\mathcal{ALC}$. As we will see in Section 4, Q-satisfiability can be decided in PTIME, and thus Theorem 7 yields a tight complexity bound for (pure) reasoning with $\mathcal{ALC}(\mathsf{Q})$.

3.2 Standard Extensions

We now discuss the extension of $\mathcal{ALC}(\mathcal{D})$ with several means of expressivity that can be considered "standard" in the area of description logics. Let us start with general TBoxes.

General TBoxes

In [Baader & Hanschke 1992], it is proved that $\mathcal{ALC}(\mathsf{R})$ extended with a transitive closure constructor on roles (similar to the star-operator of propositional dynamic logic) is undecidable, where R is a concrete domain based on Tarski algebra. The undecidability proof, which uses a reduction of the Post Correspondence Problem (PCP), can easily be adapted to $\mathcal{ALC}(\mathsf{R})$ extended with general TBoxes, which is thus also undecidable. This adaption is performed in [Lutz 2001b; 2002c], where not only R is considered, but a more general result is obtained that applies to a large class of concrete domains.

THEOREM 8 *For concrete domains $\mathcal{D}$ such that (i) $\mathbb{N} \subseteq \Delta_{\mathcal{D}}$ and (ii) $\Phi_{\mathcal{D}}$ provides a unary predicate for equality with 0, a binary equality predicate, and a binary predicate for incrementation, $\mathcal{ALC}(\mathcal{D})$-concept satisfiability and subsumption w.r.t. general TBoxes are undecidable.*

Note that there exist rather simple (and admissible) concrete domains satisfying the conditions listed in the theorem, an example being the concrete domain Q.† Since Q-satisfiability can be decided in PTIME (Section 4), it should be clear that the reason for undecidability is an interaction between general TBoxes and concrete domains and *not* reasoning with arithmetic concrete domains themselves.

Since general TBoxes play a very important role in most application areas and are provided by almost all state-of-the-art description logics, the above result is rather discouraging. There are two ways for regaining decidability: either use a less powerful concrete domain constructor or very carefully choose the concrete domains used.

The first approach was adopted in [Haarslev *et al.* 2001] and [Horrocks & Sattler 2001]. In the former article, Haarslev et al. propose to allow only concrete features inside the concrete domain constructor instead of paths

† Strictly speaking, Q does not contain a predicate for addition with 1, but this is compensated by the predicates "$=_1$" and $+$.

of arbitrary length. In the following, we call concepts satisfying this condition *path-free*. More precisely, Haarslev et al. introduce the rather powerful description logic $\mathcal{SHN}(\mathcal{D})^{\ddagger}$, which extends $\mathcal{ALC}(\mathcal{D})$ with expressive means such as unqualified number restrictions (a weak form of graded modalities) and role hierarchies (TBox-like assertions that allow to state inclusions between roles). If path-freeness is not assumed, then $\mathcal{SHN}(\mathcal{D})$-concept satisfiability and subsumption is undecidable since reasoning with general TBoxes can be reduced to reasoning without general TBoxes—the so-called "internalization of TBoxes", c.f. [Schild 1991; Horrocks & Sattler 1999]. However, using a tableau algorithm Haarslev et al. [2001] were able to show the following:

THEOREM 9 (Haarslev et al.) *If the concrete domain $\mathcal{D}$ is admissible, then path-free $\mathcal{SHN}(\mathcal{D})$-concept satisfiability and subsumption w.r.t. general TBoxes are decidable.*

Horrocks and Sattler [2001] propose to admit only unary concrete domain predicates to overcome undecidability. Under this restriction, they prove decidability of reasoning with the very expressive description logic $\mathcal{SHOQ}(\mathcal{D})$ and general TBoxes by devising an appropriate tableau algorithm. However, allowing only unary predicates is strictly less expressive than requiring path-freeness: the concept $\exists f_1 \cdots f_k g.P$ (with P unary predicate) can clearly be replaced with the equivalent one $\exists f_1.\exists f_2.\cdots.\exists f_k.\exists g.P$ that does not use paths of length greater than one. In [Pan & Horrocks 2002], the initial result is strengthened by admitting concrete domain predicates of arbitrary arity, adopting path-freeness, and adding some additional means of expressivity (see Section 3.3). The resulting DL is called $\mathcal{SHOQ}(\mathcal{D}_n)$.

THEOREM 10 (Horrocks, Pan, Sattler) *If the concrete domain $\mathcal{D}$ is admissible, then path-free $\mathcal{SHOQ}(\mathcal{D}_n)$-concept satisfiability and subsumption w.r.t. general TBoxes are decidable.*

A more general result has been obtained in Section 5.3 of [Baader *et al.* 2002b], where it is shown that any description logic $\mathcal{L}$, such that (i) $\mathcal{L}$-concept satisfiability w.r.t. general TBoxes is decidable and (ii) $\mathcal{L}$ is "closed under disjoint unions" (see [Baader *et al.* 2002b] for details), can be extended with the path-free variant of the concrete domain constructor without losing decidability of reasoning with general TBoxes. This result generalizes Theorem 9 but not Theorem 10 since $\mathcal{SHOQ}$ does not satisfy Property (ii). Indeed, the "harmlessness" of the path-free concrete domain constructor is not very surprising since dropping paths deprives concrete do-

‡This logic is also called $\mathcal{ALCNH}_{R^+}(\mathcal{D})$.

mains of most of their expressive power: in Section 2.4.1 of [Lutz 2002b], it is shown that the path-free variant of the concrete domain constructor can be "simulated" by concept names, which is not possible for the variant admitting paths of arbitrary length. In the same section, it is proved that path-free $\mathcal{ALC}(\mathcal{D})$-concept satisfiability and subsumption w.r.t. general TBoxes are ExpTime-complete if $\mathcal{D}$ is admissible and $\mathcal{D}$-satisfiability is in ExpTime.

We now discuss the second approach to overcome undecidability of $\mathcal{ALC}(\mathcal{D})$ with general TBoxes, namely to keep the original version of the concrete domain constructor and look for concrete domains that are both interesting and do not destroy decidability of reasoning with general TBoxes. The first positive result following this route was established in [Lutz 2001a], where a concrete domain C is considered that is based on the rational numbers $\mathbb{Q} = \Delta_{\mathsf{C}}$, and provides for the binary predicates $<, \leq, =, \neq, \geq, >$ with the obvious extension. Using an automata-based approach, the following result is obtained:

THEOREM 11 *$\mathcal{ALC}(\mathsf{C})$-concept satisfiability and subsumption w.r.t. general TBoxes are* ExpTime*-complete.*

It is then shown that this result can be extended to an interval-based, temporal concrete domain. Theorem 11 has subsequently been generalized in [Lutz 2002a]: first, the concrete domain C has been extended to C^+ which, additionally, admits unary predicates $=_q$ for each $q \in \mathbb{Q}$ (with the obvious extension). Second, the "description logic part" is extended from $\mathcal{ALC}$ to the very expressive DL $\mathcal{SHIQ}$ that plays an important role in many application areas [Horrocks *et al.* 2000]. The following theorem is proved in [Lutz 2002a], also using an automata-theoretic approach:

THEOREM 12 *$\mathcal{SHIQ}(\mathsf{C}^+)$-concept satisfiability and subsumption w.r.t. general TBoxes are* ExpTime*-complete.*

Note that this logic is called $\mathbb{Q}$-$\mathcal{SHIQ}$ in [Lutz 2002a]. It is very unlikely that C^+ can be extended with any form of arithmetics without losing decidability. For example, if $\mathcal{ALC}(\mathsf{C}^+)$ is extended with a binary predicate for incrementation with one, we obtain undecidability of reasoning w.r.t. general TBoxes from Theorem 8. An interesting open question is whether a unary predicate int can be added whose extension are the integers. Such a predicate would be very useful for many applications.

Acyclic TBoxes

If we restrict ourselves to acyclic TBoxes rather than admitting general ones, the situation becomes much simpler: it is well-known that concept satisfiability w.r.t. acyclic TBoxes can be reduced to concept satisfiability without TBoxes by using *unfolding* [Nebel 1990]: given an input concept C and an acyclic TBox $\mathcal{T}$, we can exhaustively replace concept names in C that appear on the left-hand side of a concept definition in $\mathcal{T}$ with the corresponding right-hand side. This process terminates since $\mathcal{T}$ is acyclic. Moreover, it is not hard to see that the resulting concept is satisfiable iff C is satisfiable w.r.t. $\mathcal{T}$. Thus, Theorem 6 implies that $\mathcal{ALC}(\mathcal{D})$-concept satisfiability and subsumption w.r.t. acyclic TBoxes are decidable if $\mathcal{D}$ is admissible—although unfolding involves an exponential blow-up in size.

Concerning complexity, the results obtained for reasoning with $\mathcal{ALC}(\mathcal{D})$ and acyclic TBoxes are much more surprising: it is well-known that, for almost all description logics considered in the literature, adding acyclic TBoxes does not increase the complexity of reasoning. For example, $\mathcal{ALC}$-concept satisfiability and subsumption are PSpace-complete, both with and without acyclic TBoxes [Schmidt-Schauß & Smolka 1991; Lutz 1999]. Interestingly, this is not the case for $\mathcal{ALC}(\mathcal{D})$: although pure $\mathcal{ALC}(\mathcal{D})$-concept satisfiability is PSpace-complete, in [Lutz 2001b; 2002c] a large class of so-called *arithmetic* concrete domains $\mathcal{D}$ is identified for which $\mathcal{ALC}(\mathcal{D})$-concept satisfiability w.r.t. acyclic TBoxes is considerably harder, namely NExpTime-complete.

DEFINITION 13 (Arithmetic) *A concrete domain $\mathcal{D}$ is called* arithmetic *iff $\Delta_{\mathcal{D}}$ contains the natural numbers and $\Phi_{\mathcal{D}}$ contains*

- *unary predicates for equality with zero and with one,*
- *a binary equality predicate, and*
- *ternary predicates expressing addition and multiplication.*

A NExpTime-complete variant of the Post Correspondence Problem is used to show the following result:

THEOREM 14 *For any arithmetic concrete domain $\mathcal{D}$, $\mathcal{ALC}(\mathcal{D})$-concept satisfiability w.r.t. acyclic TBoxes is* NExpTime*-hard.*

Since concept satisfiability can be reduced to *non*-subsumption, this implies a co-NExpTime lower bound for $\mathcal{ALC}(\mathcal{D})$-concept subsumption if $\mathcal{D}$ is arithmetic. A corresponding upper bound is established using a tableau algorithm:

THEOREM 15 *$\mathcal{ALC}(\mathcal{D})$-concept satisfiability w.r.t. acyclic TBoxes is in* NExpTime *if $\mathcal{D}$ is admissible and $\mathcal{D}$-satisfiability is in* NP.

Again, we have to consider the complementary complexity class for subsumption. It is interesting that the addition of the seemingly harmless acyclic TBoxes results in a leap of complexity from PSpace-completeness to NExpTime-completeness.

Concept- and Role-Constructors

Interestingly, acyclic TBoxes are not the only means of expressivity that considerably increases the complexity of reasoning if added to $\mathcal{ALC}(\mathcal{D})$. In [Lutz 2001b; 2002c; Areces & Lutz 2002], analogues of Theorems 14 and 15 have been proved for the following extensions of $\mathcal{ALC}(\mathcal{D})$:

– Inverse roles. We can now additionally use expressions R^- inside the $\exists R.C$ and $\forall R.C$ constructors, where R may also be an abstract feature. The interpretation $(R^-)^\mathcal{I}$ of R^- is obtained by taking the converse of the relation $R^\mathcal{I}$. Inverses of abstract (or even concrete) features inside the concrete domain constructor are *not* allowed since the inverse of a feature is not necessarily functional.

– Role conjunction. We admit roles like $R_1 \sqcap \cdots \sqcap R_n$ inside the $\exists R.C$ and $\forall R.C$ constructors, where the R_i may also be abstract features. The interpretation $(R_1 \sqcap \cdots \sqcap R_n)^\mathcal{I}$ of $R^\mathcal{I}$ is obtained by taking the intersection of the relations $R_1^\mathcal{I}, \ldots, R_n^\mathcal{I}$. Conjunctions of abstract (or even concrete) features inside the concrete domain constructor are *not* allowed.

– Nominals. Nominals (known, e.g., from hybrid logic [Areces & de Rijke 2001]) are a new syntactic type that is used in the same way as concept names, but interpreted in *singleton* sets.

All these means of expressivity (with the possible exception of nominals) are usually considered “harmless” w.r.t. complexity, i.e., in most cases they do not increase the complexity of reasoning when added to a description logic. The above results thus show that the PSpace upper complexity bound for reasoning with $\mathcal{ALC}(\mathcal{D})$ is *not robust*, but rather quite unstable w.r.t. extensions of the language.

Although formal proofs are missing, most other standard means of expressivity are very likely to preserve decidability and the PSpace upper bound when added to $\mathcal{ALC}(\mathcal{D})$. Such means of expressivity are, e.g., qualified number restrictions (the DL counterpart of graded modalities) and transitive roles (i.e., a new sort of role names interpreted in transitive relations—not to be confused with the transitive closure role constructor). Concerning decidability, some results can be obtained by using the transfer results for fusions of description logics presented in [Baader *et al.* 2002b]. This is to some extent discussed in Section 5.6 of [Lutz 2002b], where the following result is obtained:

THEOREM 16 *If $\mathcal{D}$ is admissible, then pure $\mathcal{ALCQ}^-_{R^+}(\mathcal{D})$-concept satisfiability is decidable.*

Here, $\mathcal{ALCQ}^-_{R^+}(\mathcal{D})$ is $\mathcal{ALC}(\mathcal{D})$ extended with inverse roles, qualifying number restrictions, and transitive roles. It should also be noted that concrete domains can be combined with so-called feature agreements and disagreements without spoiling the PSpace upper complexity bound [Lutz 2002d].

3.3 Concrete Domain-Related Extensions

We now review various proposals for enhancing the expressive power of $\mathcal{ALC}(\mathcal{D})$ by extending the "concrete domain part" of this logic.

Generalized Concrete Domain Constructor

In the original version of $\mathcal{ALC}(\mathcal{D})$ as defined in Section 2, we only allow abstract features to be used in the concrete domain concept constructor instead of admitting arbitrary role names. This observation leads to a natural generalization of the concrete domain constructor that has first been proposed by Hanschke [1992].

DEFINITION 17 ($\mathcal{ALCP}(\mathcal{D})$) *A sequence $U = R_1 \cdots R_k g$ where $R_1, \ldots, R_k \in \mathsf{N_R}$ ($k \geq 0$) and $g \in \mathsf{N_{cF}}$ is called a* role path. *For an interpretation $\mathcal{I}$, $U^\mathcal{I}$ is defined as*

$$\{(d,x) \subseteq \Delta_\mathcal{I} \times \Delta_\mathcal{D} \mid \exists d_1, \ldots, d_{k+1} : d = d_1, \\ (d_i, d_{i+1}) \in R_i^\mathcal{I} \text{ for } 1 \leq i \leq k, \text{ and } g^\mathcal{I}(d_{k+1}) = x\}.$$

$\mathcal{ALCP}(\mathcal{D})$ is obtained from $\mathcal{ALC}(\mathcal{D})$ by allowing the use of concepts of the form $\forall U_1, \ldots, U_n.P$ and $\exists U_1, \ldots, U_n.P$ in place of concept names, where $P \in \Phi_\mathcal{D}$ is of arity n and $U_1, \ldots, U_n$ are role paths. The semantics of the generalized concrete domain constructors is defined as follows:

$$(\forall U_1, \ldots, U_n.P)^\mathcal{I} := \{d \in \Delta_\mathcal{I} \mid \text{ For all } x_1, \ldots, x_n \text{ with } (d, x_i) \in U_i^\mathcal{I}, \\ \text{we have } (x_1, \ldots, x_n) \in P^\mathcal{D}\}$$

$$(\exists U_1, \ldots, U_n.P)^\mathcal{I} := \{d \in \Delta_\mathcal{I} \mid \text{ There exist } x_1, \ldots, x_n \text{ with } (d, x_i) \in U_i^\mathcal{I} \\ \text{and } (x_1, \ldots, x_n) \in P^\mathcal{D}\}$$

Obviously, every path is also a role path. Hence, the $\exists U_1, \ldots, U_n.P$ constructor of $\mathcal{ALCP}(\mathcal{D})$ is a generalization of the $\exists u_1, \ldots, u_n.P$ constructor of $\mathcal{ALC}(\mathcal{D})$. For paths $u_1, \ldots, u_n$, the $\mathcal{ALCP}(\mathcal{D})$-concept $\forall u_1, \ldots, u_n.P$ is equivalent to the $\mathcal{ALC}(\mathcal{D})$-concept $u_1{\uparrow} \sqcup \cdots \sqcup u_n{\uparrow} \sqcup \exists u_1, \ldots, u_n.P$. This is the reason why $\mathcal{ALC}(\mathcal{D})$ does not provide for a counterpart of the $\forall U_1, \ldots, U_n.P$ constructor.

Using the generalized constructors, we can, for example, express that the duration of subprocesses is bounded by the duration of the mother process *without committing to a particular number of subprocesses*:

$$\mathsf{Process} \sqcap \forall(\mathsf{duration}), (\mathsf{subproc}\ \mathsf{duration}).\leq,$$

where $\mathsf{duration}$ is a concrete feature. The existential version of the generalized concrete domain constructor can then be used to express that there exists a subprocess whose duration is strictly shorter than the duration of the mother process:

$$\mathsf{Process} \sqcap \exists(\mathsf{duration}), (\mathsf{subproc}\ \mathsf{duration}).<. \qquad (*)$$

Note, however, that it is now impossible to state that the subprocess with the shorter duration is a $\mathsf{DangerousProcess}$. This observation suggests that role hierarchies are a useful extension of $\mathcal{ALCP}(\mathcal{D})$: in the resulting DL, we can modify $(*)$ by replacing the role $\mathsf{subproc}$ with an abstract feature $\mathsf{specialSubprocess}$, adding the conjunct $\exists \mathsf{specialSubprocess}.\mathsf{DangerousProcess}$, and finally using a role hierarchy to state that $\mathsf{specialSubprocess}$ is a subrole of $\mathsf{subproc}$, i.e. that we have $\mathsf{specialSubProcess}^{\mathcal{I}} \subseteq \mathsf{subproc}^{\mathcal{I}}$. In the following, however, we will stick with the original variant of $\mathcal{ALCP}(\mathcal{D})$ that does not admit role hierarchies.

As shown in [Hanschke 1992], satisfiability and subsumption of $\mathcal{ALCP}(\mathcal{D})$-concepts are decidable if $\mathcal{D}$ is admissible. However, when investigating the complexity of $\mathcal{ALCP}(\mathcal{D})$, it becomes clear that initially restricting ourselves to abstract features inside the concrete domain constructor is a sensible idea since it allows a more fine-grained complexity analysis: it is shown in [Lutz 2002b] that, while reasoning with $\mathcal{ALC}(\mathcal{D})$ is PSPACE-complete, reasoning with $\mathcal{ALCP}(\mathcal{D})$ is much harder. Indeed, the complexity of (pure) reasoning with $\mathcal{ALCP}(\mathcal{D})$ parallels the complexity of reasoning with $\mathcal{ALC}(\mathcal{D})$ extended with acyclic TBoxes. The lower bound is determined by reduction of a NEXPTIME-complete variant of the PCP:

THEOREM 18 *For any arithmetic concrete domain $\mathcal{D}$, pure $\mathcal{ALCP}(\mathcal{D})$-concept satisfiability is* NEXPTIME-*hard.*

It is interesting to note that this lower bound does even hold if abstract features are dropped from the language. As in the case of acyclic TBoxes, there exists a corresponding upper bound which is established using a tableau algorithm:

THEOREM 19 *Pure $\mathcal{ALCP}(\mathcal{D})$-concept satisfiability is in* NExpTime *if $\mathcal{D}$ is admissible and $\mathcal{D}$-satisfiability is in* NP.

We obtain corresponding co-NExpTime complexity bounds for concept subsumption. Another generalization of the concrete domain constructor has been proposed in [Pan & Horrocks 2002]: the authors replace concrete features by concrete roles, which are not required to be functional. Additionally, they allow the application of number restrictions to concrete roles. This allows, for example, to state that each person has exactly one age (attached via a concrete role age) while being allowed to have many telephone numbers (attached via a concrete role tel).

A Concrete Domain Role Constructor

Another natural extension of the original variant of $\mathcal{ALC}(\mathcal{D})$ is obtained by using the concrete domain not only to define concepts, but by additionally allowing the definition of complex roles with reference to concrete domain predicates. Such an extension has first been proposed in [Haarslev *et al.* 1999].

DEFINITION 20 ($\mathcal{ALCRP}(\mathcal{D})$) *A* concrete domain role *is an expression of the form*

$$\exists(u_1, \ldots, u_n), (v_1, \ldots, v_m).P$$

where $u_1, \ldots, u_n$ and $v_1, \ldots, v_m$ are paths and P is an $n+m$-ary predicate. The semantics is given as follows:

$$\begin{aligned}&(\exists(u_1, \ldots, u_n), (v_1, \ldots, v_m).P)^{\mathcal{I}} := \\ &\qquad \{(d, e) \in \Delta_{\mathcal{I}} \times \Delta_{\mathcal{I}} \mid \textit{There exist } x_1, \ldots, x_n \textit{ and } y_1, \ldots, y_m \\ &\qquad\ \textit{such that } u_i^{\mathcal{I}}(d) = x_i \textit{ for } 1 \leq i \leq n, v_i^{\mathcal{I}}(e) = y_i \textit{ for } 1 \leq i \leq m, \textit{ and} \\ &\qquad\ (x_1, \ldots, x_n, y_1, \ldots, y_m) \in P^{\mathcal{D}}\}\end{aligned}$$

$\mathcal{ALCRP}(\mathcal{D})$ is obtained from $\mathcal{ALC}(\mathcal{D})$ by allowing the use of concrete domain roles inside the $\exists R.C$ and $\forall R.C$ constructors.

Note that concrete domain roles are *not* allowed inside the concrete domain concept constructor. Let us view an example $\mathcal{ALCRP}(\mathcal{D})$-concept. Assume that we use a concrete domain based on temporal intervals and binary predicates describing the possible relationships between such intervals. Then the concept

$$\mathsf{Process} \sqcap \forall(\exists(\mathsf{exectime}), (\mathsf{exectime}).\mathsf{overlaps}).\neg\mathsf{DangerousProcess},$$

describes processes that are not temporally overlapping with dangerous processes. Note that $\exists(\mathsf{exectime}), (\mathsf{exectime}).\mathsf{overlaps}$ is a concrete domain role defined in terms of the binary predicate $\mathsf{overlaps}$ and the concrete feature $\mathsf{exectime}$ which associates processes with the time interval in which they are executed.

In [Lutz & Möller 1997], a reduction of the Post Correspondence Problem is used to prove that there exist concrete domains $\mathcal{D}$ such that the satisfiability of $\mathcal{ALCRP}(\mathcal{D})$-concepts is undecidable. It is straightforward to generalize this result to the class of concrete domains identified in Theorem 8:

THEOREM 21 (Lutz, Möller) *For concrete domains $\mathcal{D}$ such that (i) $\mathbb{N} \subseteq \Delta_\mathcal{D}$ and (ii) $\Phi_\mathcal{D}$ provides a unary predicate for equality with 0, a binary equality predicate, and a binary predicate for incrementation, pure $\mathcal{ALCRP}(\mathcal{D})$-concept satisfiability and subsumption are undecidable.*

In [Haarslev *et al.* 1999] a fragment of $\mathcal{ALCRP}(\mathcal{D})$ is identified that is closed under negation, strictly extends $\mathcal{ALC}(\mathcal{D})$, and is decidable for all admissible concrete domains. To introduce this fragment, we need a way to convert $\mathcal{ALCRP}(\mathcal{D})$-concepts into equivalent ones in NNF. Assuming that $\mathcal{D}$ is admissible, this conversion can be done by eliminating double negation and using de Morgan's rules, the duality between $\exists R.C$ and $\forall R.C$, and the equivalences

$$\begin{array}{rcl}\neg(\exists u_1, \ldots, u_n.P) & \equiv & \exists u_1, \ldots, u_n.\overline{P} \sqcup u_1\uparrow \sqcup \cdots \sqcup u_n\uparrow \\ \neg(g\uparrow) & \equiv & \exists g.\top_\mathcal{D}\end{array}$$

where, for P an n-ary predicate, $\overline{P}$ denotes the predicate satisfying $\overline{P}^\mathcal{D} = \Delta_\mathcal{D}^n \setminus P^\mathcal{D}$, which exists since $\mathcal{D}$ is admissible. In the following, $\mathsf{sub}(C)$ refers to the set of subconcepts of the concept C (including C itself).

DEFINITION 22 (Restricted $\mathcal{ALCRP}(\mathcal{D})$-concept) *An $\mathcal{ALCRP}(\mathcal{D})$-concept C is called* restricted *iff the result C' of converting C to NNF satisfies the following conditions:*

1. *For any $\forall R.D \in \mathsf{sub}(C')$, where R is a concrete domain role,*
   (a) *$\mathsf{sub}(D)$ does not contain any concepts $\exists S.E$ with S a concrete domain role, and*
   (b) *if $\mathsf{sub}(D)$ contains a concept $\exists u_1, \ldots, u_n.P$, then $u_1, \ldots, u_n \in \mathsf{N_{cF}}$.*
2. *For any $\exists R.D \in \mathsf{sub}(C')$, where R is a concrete domain role,*

(a) $\mathsf{sub}(D)$ *does not contain any concepts* $\forall S.E$ *with* S *a concrete domain role, and*

(b) *if* $\mathsf{sub}(D)$ *contains a concept* $\exists u_1, \ldots, u_n.P$*, then* $u_1, \ldots, u_n \in \mathsf{N_{cF}}$.

It is easily seen that the set of restricted $\mathcal{ALCRP}(\mathcal{D})$-concepts is closed under negation. Hence, subsumption of restricted $\mathcal{ALCRP}(\mathcal{D})$-concepts can still be reduced to satisfiability of restricted $\mathcal{ALCRP}(\mathcal{D})$-concepts (and vice versa). Decidability of restricted $\mathcal{ALCRP}(\mathcal{D})$-concept satisfiability and subsumption has been shown in [Haarslev *et al.* 1999], where it is also illustrated that this fragment of $\mathcal{ALCRP}(\mathcal{D})$ is still useful for reasoning about spatio-terminological knowledge. The complexity of reasoning has been investigated in [Lutz 2002b], where it is shown that, once more, we can use a NExpTime-complete variant of the PCP and a tableau algorithm to prove the following:

THEOREM 23 *Let* $\mathcal{D}$ *be a concrete domain. If* $\mathcal{D}$ *is arithmetic, then (pure) satisfiability of restricted* $\mathcal{ALCP}(\mathcal{D})$*-concepts is* NExpTime*-hard. If* $\mathcal{D}$ *is admissible and* $\mathcal{D}$*-satisfiability is in* NP*, then (pure) satisfiability of restricted* $\mathcal{ALCRP}(\mathcal{D})$*-concepts can be decided in* NExpTime.

Again, we obtain corresponding co-NExpTime bounds for concept subsumption.

Aggregation Functions

Aggregation is a useful mechanism available in many expressive representation formalisms such as database schema and query languages. It is thus a natural idea to extend description logics providing for concrete domains with aggregation as proposed in [Baader & Sattler 2002]. Consider, for example, a process description

$$\mathsf{Process} \sqcap \exists \mathsf{duration}.{>_0} \sqcap \forall \mathsf{subproc}.(\mathsf{Process} \sqcap \exists \mathsf{duration}.{>_0}).$$

The aggregation function "sum" is needed if we want to express that the duration of the mother process is identical to the sum of the durations of all its subprocesses (of which there may be arbitrarily many).

DEFINITION 24 (Aggregation) *A* concrete domain with aggregation *is a concrete domain that, additionally, provides for a set of aggregation functions* $\mathsf{agg}(\mathcal{D})$*, where each* $\Gamma \in \mathsf{agg}(\mathcal{D})$ *is associated with a partial function* $\Gamma^{\mathcal{D}}$ *from the set of finite multisets of* $\mathsf{dom}(\mathcal{D})$ *into* $\mathsf{dom}(\mathcal{D})$.[§]

[§] Intuitively, a multiset is a set that may contain the same element multiple (but only finitely many) times.

To distinguish concrete domains with aggregation from those without, we denote the former with Σ. Typical aggregation functions are min, max, sum, count, and $\mathsf{average}$ (with the obvious extensions).

$\mathcal{ALC}(\Sigma)$-concepts are now defined in the same way as $\mathcal{ALC}(\mathcal{D})$-concepts except that *aggregated features* may be substituted for concrete features, where an aggregated feature is an expression $\Gamma(R \circ g)$ with R role, g concrete feature, and Γ an aggregation function from Σ. The semantics of aggregated features is defined via multisets:

DEFINITION 25 (Semantics of $\mathcal{ALC}(\Sigma)$) *Let $\mathcal{I}$ be an interpretation. For each $d \in \Delta_\mathcal{I}$ such that the set $\{e \mid (d,e) \in R^\mathcal{I}\}$ is finite, we use $M_d^{R \circ g}$ to denote the multiset that, for each $z \in \Delta_\mathcal{D}$, contains z exactly $|\{e \mid (d,e) \in R^\mathcal{I}$ and $g^\mathcal{I}(e) = z\}|$ times. The semantics of aggregated features is now defined as follows:*

$$\Gamma(R \circ g))^\mathcal{I}(d) := \begin{cases} \Gamma^\Sigma(M_d^{R \circ g}) & \textit{if } \{e \mid (d,e) \in R^\mathcal{I}\} \textit{ is finite} \\ \textit{undefined} & \textit{otherwise.} \end{cases}$$

Returning to the initial example, we can now express the fact that the duration of the mother process is identical to the sum of the durations of all its subprocesses by writing

$$\exists \mathsf{duration}, \mathsf{sum}(\mathsf{subproc} \circ \mathsf{duration}).{=}.$$

The investigations performed by Baader and Sattler [2002] reveal that the expressive power provided by aggregation functions is hard to tame. The following result is proved by a reduction of Hilbert's 10-th problem.

THEOREM 26 (Baader, Sattler) *For concrete domains with aggregation Σ where (i) $\mathsf{dom}(\Sigma)$ includes the non-negative integers, (ii) Φ_Σ contains a (unary) predicate for equality with 1 and a (binary) equality predicate, and (iii) $\mathsf{agg}(\Sigma)$ contains min, max, and sum, pure $\mathcal{ALC}(\Sigma)$-concept satisfiability and subsumption are undecidable.*

This lower bound does even apply if we admit only conjunction, the $\forall R.C$ constructor, and the concrete domain constructor, but drop all other concept constructors. Rather strong measures have to be taken in order to regain decidability: either, we have to drop the $\forall R.C$ constructor from the language or we have to confine ourselves with "well-behaved" aggregation functions. Following the first approach, one may replace the logic $\mathcal{ALC}(\Sigma)$ with the DL $\mathcal{EL}(\Sigma)$ that only provides for the following concept constructors: atomic negation (i.e. restricted to concept names), conjunction, disjunction, the $\exists R.C$ constructor, and the concrete domain constructor. When devising decision procedures for $\mathcal{EL}(\Sigma)$, requiring concrete domains to be admissible is no longer sufficient since the multisets underlying aggregation functions need to be dealt with: Σ-conjunctions may, additionally, contain multiset variables and inclusion statements between multisets (for a precise definition see [Baader & Sattler 2002]). If the satisfiability of such extended Σ-conjunctions is decidable, we call Σ *aggregation-admissible.* Baader and Sattler [2002] prove the following result by devising a tableau algorithm:

THEOREM 27 (Baader, Sattler) *For concrete domains with aggregation Σ that are aggregation-admissible, pure $\mathcal{EL}(\Sigma)$-concept satisfiability is decidable.*

However, subsumption of $\mathcal{EL}(\Sigma)$-concepts is, in general, still undecidable. Following the second approach, Baader and Sattler found out that only min and max can be considered well-behaved, obtaining the following result also by construction of a tableau algorithm:

THEOREM 28 (Baader, Sattler) *For concrete domains with aggregation Σ such that (i) Σ is admissible, (ii) Φ_Σ contains a binary equality predicate and a binary predicate for a linear ordering on Δ_Σ, and (iii) $\mathsf{agg}(\Sigma) = \{\min, \max\}$, pure $\mathcal{ALC}(\Sigma)$-concept satisfiability and subsumption are decidable.*

Keys

In several applications, it is useful to identify a set of concrete features whose values *uniquely* determine logical objects. Say, for example, that there exists a concrete feature socnum associating humans with their social security number. Then, if a human is American, she should be *uniquely* identified by this number. In other words, there should be no two distinct domain elements that are both in the extension of American and share the same value of the concrete feature socnum. This idea leads to the definition of key boxes, which have been proposed in [Areces *et al.* 2002].

DEFINITION 29 (Key box) *A* key box *is a finite set of* key definitions

$$(u_1, \ldots, u_n \textit{ keyfor } C),$$

where $u_1, \ldots, u_n$ are paths and C is a concept. An interpretation $\mathcal{I}$ satisfies *a key definition $(u_1, \ldots, u_n$ keyfor $C)$ iff, for any $a, b \in C^{\mathcal{I}}$,*

$$u_i^{\mathcal{I}}(a) = u_i^{\mathcal{I}}(b) \textit{ for } 1 \leq i \leq n \textit{ implies } a = b.$$

$\mathcal{I}$ is a model *of a key box $\mathcal{K}$ iff $\mathcal{I}$ satisfies all key definitions in $\mathcal{K}$.*

Clearly, key boxes are a natural choice in database applications such as the one described in Section 1: they correspond to so-called functional dependencies which are the most popular type of constraint for relational databases. For this reason, keys for description logics have also been considered in a non-concrete domain context [Borgida & Weddell 1997; Calvanese *et al.* 2000; Khizder *et al.* 2001].

From a logical perspective, there exists a close relationship between nominals and key boxes. For example, if used together with the key definition (g keyfor $\top$), then the $\mathcal{ALC}(\mathsf{Q})$-concept $\exists g.{=_q}$ "behaves" like a nominal for each $q \in \mathbb{Q}$: it is interpreted either in the empty set or in a singleton set.

Indeed, key boxes are a quite powerful expressive means. This is reflected by the computational complexity of $\mathcal{ALCK}(\mathcal{D})$, the extension of $\mathcal{ALC}(\mathcal{D})$ with key boxes, which is investigated in [Areces *et al.* 2002]. The following undecidability result is proved by a reduction of the PCP:

THEOREM 30 (Areces et al.) *For any arithmetic concrete domain $\mathcal{D}$, pure $\mathcal{ALCK}(\mathcal{D})$-concept satisfiability and subsumption w.r.t. key boxes are undecidable.*

Decidability can be regained by allowing only Boolean combinations of concept names on the right-hand side of key definitions. Key boxes satisfying this property are called *Boolean.* Pure $\mathcal{ALCK}(\mathcal{D})$-concept satisfiability and subsumption w.r.t. Boolean key boxes are NExpTime-hard for arithmetic concrete domains $\mathcal{D}$. Surprisingly, [Areces *et al.* 2002] can even show that this high complexity cannot be reduced if paths are restricted to length one inside $\mathcal{ALCK}(\mathcal{D})$-concepts and key boxes. In analogy to Section 3.2, where this approach helped to overcome undecidability in the presence of general TBoxes, we call such concepts and key boxes *path-free.* The following theorem is proved by reduction of a NExpTime-complete variant of the PCP:

THEOREM 31 (Areces et al.) *For any arithmetic concrete domain $\mathcal{D}$, pure path-free $\mathcal{ALCK}(\mathcal{D})$-concept satisfiability and subsumption w.r.t. Boolean and path-free key boxes are* NExpTime*-hard.*

To devise a decision procedure for reasoning with key boxes, it does not suffice to assume admissibility of concrete domains: the concrete domain reasoner should not only tell us whether a given $\mathcal{D}$-conjunction is satisfiable, but also which variables in it must take the same value in solutions.

DEFINITION 32 (key-admissible) *A concrete domain $\mathcal{D}$ is called* key-admissible *iff there exists an algorithm that takes as input a $\mathcal{D}$-conjunction c, returns* clash *if c is unsatisfiable, and otherwise non-deterministically outputs an equivalence relation $\sim$ on the set of variables V used in c such that there exists a solution δ for c with the following property:*

$$\delta(v) = \delta(v') \text{ iff } v \sim v' \text{ for all } v, v' \in V.$$

We say that extended $\mathcal{D}$-satisfiability is in NP *if there exists an algorithm as above running in polynomial time.*

This property is much less esoteric than it seems: as noted in [Areces *et al.* 2002], any concrete domain that is admissible and provides for an equality predicate is also key-admissible. This rather weak condition is satisfied by almost all (admissible) concrete domains proposed in the literature, c.f. Section 4. Using a tableau algorithm, Areces et al. [2002] obtain a matching upper bound for Theorem 31:

THEOREM 33 (Areces et al.) *Let $\mathcal{D}$ be a concrete domain that is key-admissible. If extended $\mathcal{D}$-satisfiability is in* NP*, then pure $\mathcal{ALCOK}(\mathcal{D})$-concept satisfiability w.r.t. Boolean key boxes is in* NExpTime.

Note that, in contrast to Theorem 31, concepts do not have to be path-free. As usual, corresponding co-NExpTime results are obtained for concept subsumption.

Areces et al. also consider the extension of the description logic $\mathcal{SHOQ}(\mathcal{D}_n)$ (see Section 3.2) with key boxes. Since $\mathcal{SHOQ}(\mathcal{D}_n)$ provides only for the path-free variant of the concrete domain constructor, it is natural to require key boxes to also be path-free. Due to the fact that each path-free $\mathcal{ALCK}(\mathcal{D})$-concept is also a path-free $\mathcal{SHOQ}(\mathcal{D}_n)$-concept, Theorem 31 provides us with a lower NExpTime complexity bound. In [Areces *et al.* 2002], the corresponding upper bound is obtained by devising an appropriate tableau algorithm:

THEOREM 34 (Areces et al.) *Let $\mathcal{D}$ be a concrete domain that is key-admissible. If extended $\mathcal{D}$-satisfiability is in* NP*, then path-free $\mathcal{SHOQ}(\mathcal{D}_n)$-concept satisfiability w.r.t. path-free key boxes is in* NExpTime.

Note that the key boxes in Theorem 34 are not required to be Boolean! We obtain a corresponding co-NExpTime bound for concept subsumption.

4 Concrete Domains

In this section, we discuss several concrete domains that have been proposed in the literature. We start with numerical concrete domains, which are useful in a wide range of application areas, and then consider more specific concrete domains such as temporal and spatial ones.

4.1 Numerical Concrete Domains

Let us start with reconsidering the concrete domain Q introduced in Section 2. It is based on the rational numbers $\mathbb{Q}$ and provides for the following predicates:

- (unary) predicates $<_q$, $\leq_q$, $=_q$, $\neq_q$, $\geq_q$, and $>_q$ for comparisons with rational numbers q;
- binary comparison predicates $<$, $\leq$, $=$, $\neq$, $\geq$, and $>$;
- a ternary addition predicate $+$ and its negation $\overline{+}$;
- unary predicates $\top_{\mathsf{Q}}$ and $\bot_{\mathsf{Q}}$ (for admissibility).

Note that we could drop some of the predicates since, e.g., $\exists u.{<_7}$ can be written as $\exists g.{=_7} \sqcap \exists u, g.{<}$, and $\exists u_1, u_2.{\geq}$ can be written as $\exists u_1, u_2.{=} \sqcup \exists u_2, u_1.{<}$.[¶] It is not very hard to prove that Q-satisfiability is in PTime using a reduction to linear programming (LP). More precisely, a *linear programming problem* has the form $Ax = b$, where A is an $m \times n$-matrix of rational numbers, x is an n-vector of variables, and b is an m-vector of rational numbers (see, e.g. [Schrijver 1986]). A *solution* of $Ax = b$ is a mapping δ that assigns a rational number to each variable such that the equality $Ax = b$ holds. Deciding whether a given LP problem has a solution is well known to be in PTime [Schrijver 1986]. Details on the reduction of Q-satisfiability to linear programming can be found in [Lutz 2002d].

There exist several interesting predicates that can be added to Q in order to extend its expressive power. From the viewpoint of many applications, the most useful ones are the following:

- ternary predicates $*$ and $\overline{*}$ with $(*)^{\mathsf{Q}} = \{(q, q', q'') \in \mathbb{Q}^3 \mid q \cdot q' = q''\}$ and $(\overline{*})^{\mathsf{Q}} = \mathbb{Q}^3 \setminus (*)^{\mathsf{Q}}$;
- unary predicates int and $\overline{\mathsf{int}}$ with $(\mathsf{int})^{\mathsf{Q}} = \mathbb{Z}$ (where $\mathbb{Z}$ denotes the integers) and $(\overline{\mathsf{int}})^{\mathsf{Q}} = \mathbb{Q} \setminus \mathbb{Z}$.

[¶] It is sufficient to provide, for example, the predicates $\{=_q \mid q \in \mathbb{Q}\} \cup \{<, +\}$: all other predicates can be defined in terms of these. The corresponding concrete domain is, however, not closed under negation and thus not admissible.

	Complexity of $\mathcal{D}$-satisfiability	Complexity of (pure) $\mathcal{ALC}(\mathcal{D})$-concept satisfiability
Q + '$*$' + 'int'	undecidable	undecidable
Q + '$*$'	in ExpTime	in NExpTime
Q + 'int'	NP-complete	PSpace-complete
Q	in PTime	PSpace-complete

Figure 1. Numerical concrete domains and their complexity.

Adding different combinations of these predicates, we obtain three extensions of the concrete domain Q, which are listed in Figure 1, together with known complexity bounds on $\mathcal{D}$-satisfiability and pure $\mathcal{ALC}(\mathcal{D})$-concept satisfiability. Note that, since the obtained concrete domains should be admissible, we assume that the addition of the predicates '$*$' and 'int' implies the addition of their negations. Let us discuss the given bounds in some more detail:

- It is easily proved that the concrete domain Q + '$*$' + 'int' is undecidable using a reduction of Hilbert's 10-th problem. Clearly, the undecidability is inherited by $\mathcal{ALC}(\mathcal{D})$-concept satisfiability.

- The ExpTime upper bound for Q + '$*$' stems from the fact that, for this concrete domains, finite predicate conjunctions can be translated into formulas of Tarski algebra (also known as the theory of real closed fields) without quantifier alternation. The satisfiability of such formulas has been proved to be decidable in ExpTime [Mayr & Meyer 1982; Grigorev 1988]. The NExpTime upper bound for $\mathcal{ALC}(\mathcal{D})$-concept satisfiability stems from a more general variant of Theorem 7 that is proved in [Lutz 2002d].

  In [Baader & Hanschke 1992], it has even been proposed to use all formulas of Tarski algebra (also those with quantifiers) as concrete domain predicates. For the concrete domain obtained in this way, $\mathcal{D}$-satisfiability is ExpSpace-complete [Mayr & Meyer 1982].

- Finally, NP-completeness of the concrete domain Q + 'int' can be shown via mutual reductions to and from mixed integer programming (MIP), i.e., linear programming with an additional type of variables that must take integer values in solutions. Deciding the existence of a solution for MIP problems is known to be NP-complete. More details on the reductions can be found in [Lutz 2002d]. The complexity of $\mathcal{ALC}(\mathcal{D})$-concept satisfiability is then obtained from Theorem 7.

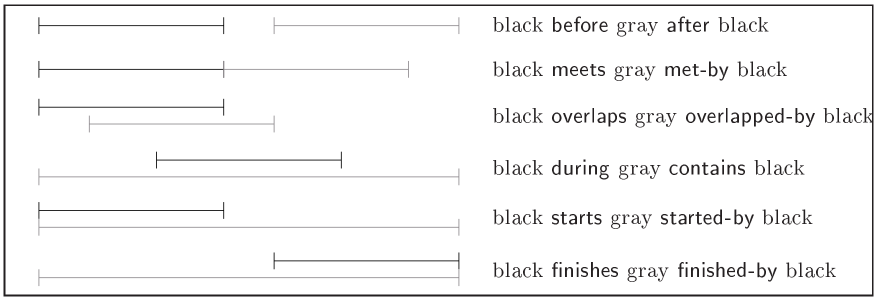

Figure 2. The Allen relations (without equal).

Note that some *fragments* of the concrete domain Q are also interesting, examples being the concrete domains C and C^+ discussed in Section 3.2: in contrast to Q itself, they can be combined with general TBoxes without losing decidability.

4.2 Other Concrete Domains

In this section, we present two examples for non-numerical concrete domains that have been proposed in the literature. The first example is concerned with representing time: since it is a natural idea to take into account temporal aspects when reasoning about conceptual knowledge, many temporal extensions of description logics have been proposed, see e.g. [Schild 1993; Artale & Franconi 1998; Wolter & Zakharyaschev 1999] and the survey [Artale & Franconi 2001]. As discussed in [Lutz 2002d; 2002b], one possible approach for such an extension is to use an appropriate, temporal concrete domain which we introduce in the following.

In temporal reasoning, one of the most fundamental decisions to be made is whether to use time points or time intervals as the atomic temporal entity. Time points can obviously be represented using numerical concrete domains such as those from Section 4.1. If we choose time intervals as our atomic temporal entity, it seems appropriate to define an interval-based, temporal concrete domain. Usually, such concrete domains are based on the 13 *Allen relations*, which describe the possible relationships between any two intervals over some temporal structure. We refer to [Allen 1983] for an exact definition and confine ourselves with the graphical presentation of the relations given in Figure 2. The most important property of the Allen relations is that they are jointly exhaustive and pairwise disjoint, i.e., for each temporal structure $(T, \prec)$ and $t_1, t_2 \in T$, there exists exactly one relation r such that $t_1 \, \mathsf{r} \, t_2$. We now define a concrete domain I that is based on the temporal structure $(\mathbb{Q}, <)$:

- $\Delta_\mathrm{I} := \{(t_1, t_2) \mid t_1, t_2 \in \mathbb{Q} \text{ and } t_1 < t_2\}$;
- Φ_I contains unary predicates $\top_\mathrm{I}$ and $\bot_\mathrm{I}$ (with the obvious extension) and binary predicates rel and $\overline{\mathsf{rel}}$ for each Allen relation rel such that $(\mathsf{rel})^\mathrm{I} = \{(i_1, i_2) \in \Delta_\mathrm{I} \times \Delta_\mathrm{I} \mid i_1 \ \mathsf{rel} \ i_2\}$ and $(\overline{\mathsf{rel}})^\mathrm{I} = \Delta_\mathrm{I}^2 \setminus (\mathsf{rel})^\mathrm{I}$.

It is easily verified that I satisfies Property 1 of admissibility. Moreover, it follows from standard results in temporal reasoning that I-satisfiability is NP-complete—more details can be found in Section 2.4.3 of [Lutz 2002b]. Thus, from Theorem 7 we obtain that $\mathcal{ALC}(\mathrm{I})$-concept satisfiability is PSpace-complete.

Interestingly, there exists a polynomial reduction of $\mathcal{ALC}(\mathrm{I})$-concept satisfiability to $\mathcal{ALC}(\mathsf{C})$-concept satisfiability, where C is the concrete domain based on $\mathbb{Q}$ and the binary comparisons $<, \leq, =, \neq, \geq, >$ introduced in Section 3.2: intuitively, it is possible to represent intervals in terms of their endpoints and Allen relations in terms of comparisons between interval endpoints—details can again be found in Section 2.4.3 of [Lutz 2002b]. Since this reduction also works in the presence of general TBoxes, Theorem 11 implies that $\mathcal{ALC}(\mathrm{I})$-concept satisfiability w.r.t. general TBoxes is decidable (and ExpTime-complete). The usefulness of $\mathcal{ALC}(\mathrm{I})$ with general TBoxes for temporal reasoning is illustrated in [Lutz 2001a] in a process engineering context. It should be noted that $\mathcal{ALC}(\mathrm{I})$ (without TBoxes) has been used to obtain complexity results for an interval-based temporal description logic that is not based on concrete domains [Artale & Lutz 1999].

Although spatial aspects are as important for conceptual reasoning as are temporal aspects, until now only rather few spatial description logics have been proposed, see e.g. [Haarslev *et al.* 1999; Kutz *et al.* 2001; 2002]. This is particularly surprising since, in the spatial case, the number of choices for atomic entities and relations/predicates is much larger than in the temporal case: as spatial primitives, we may use points in a metric, Euclidean, or topological space, sets of such points (to represent regions), or sets of such points with certain characteristics such as connectedness or definability by polytopes. For the predicates, we may for example choose distance relations, orientation relations, or the so-called RCC-8 relations. In the following, we take a closer look at the last possibility since this approach has been used in the only spatial description logic based on concrete domains that has been proposed in the literature [Haarslev *et al.* 1999].

The set of so-called RCC-8 relations is well-known from the area of qualitative spatial reasoning [Randell *et al.* 1992; Bennett 1997; Renz & Nebel 1999]. RCC-8 consists of eight jointly exhaustive and pairwise disjoint relations that describe the possible relationships between any two regular closed

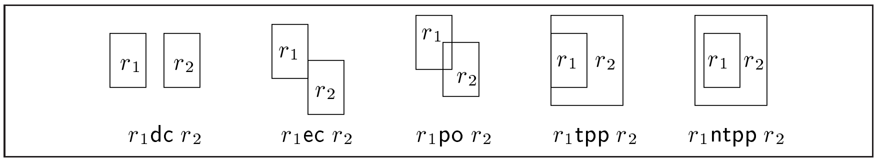

Figure 3. The RCC-8 relations in two-dimensional space.

regions in a topological space.‖ For 2D space, the relations are illustrated in Figure 3, where the equality relation eq, the inverse tppi of tpp, and the inverse ntppi of ntpp have been omitted. We now define a spatial concrete domain S based on the standard topology of two-dimensional space:

- Δ_{S} is the set $\mathcal{RC}_{\mathbb{R}^2}$ of all regular closed subsets of $\mathbb{R}^2$;
- Φ_{S} contains unary predicates $\top_{\mathsf{S}}$ and $\bot_{\mathsf{S}}$ and binary predicates rel and $\overline{\mathsf{rel}}$ for each topological relation rel such that $(\mathsf{rel})^{\mathsf{S}} = \{(r_1, r_2) \in \mathcal{RC}_{\mathbb{R}^2} \times \mathcal{RC}_{\mathbb{R}^2} \mid r_1 \ \mathsf{rel} \ r_2\}$ and $(\overline{\mathsf{rel}})^{\mathsf{S}} = \Delta_{\mathsf{S}}^2 \setminus (\mathsf{rel})^{\mathsf{S}}$..

The concrete domain S obviously satisfies Condition 1 of admissibility. Using standard results from qualitative spatial reasoning, it is straightforward to show that S-satisfiability is in NP—details can be found in [Lutz 2002d]. Thus, $\mathcal{ALC}(\mathsf{S})$-concept satisfiability is PSpace-complete by Theorem 7. In [Haarslev *et al.* 1999], the concrete domain S has been used in the description logic $\mathcal{ALCRP}(\mathsf{S})$, i.e. $\mathcal{ALC}(\mathsf{S})$ extended with the concrete domain role constructor from Section 3.3, to reason about spatio-terminological knowledge. By Theorem 23, $\mathcal{ALCRP}(\mathsf{S})$-concept satisfiability is in NExpTime—the corresponding lower bound does not apply since S is not arithmetic. It is an interesting open question whether the description logic $\mathcal{ALC}(\mathsf{S})$ can be combined with general TBoxes without losing decidability.

Other sets of relations from the area of qualitative spatial reasoning (see e.g. [Stock 1997]) may be used to define different spatial concrete domains. Interesting related work has been presented in [Kutz *et al.* 2001; 2002]: the authors propose to combine description logics with modal logics for metric spaces. The expressive power of the resulting spatial description logics seems to be orthogonal to the expressive power of spatial description logics based on concrete domains.

5 Final Remarks

In this paper, we have given an overview over the research on description logics with concrete domains, focussing on decidability and complexity re-

‖A region r is regular closed if it satisfies $ICr = r$, where C is the topological closure operator and I is the topological interior operator.

sults. We have tried to cover most relevant results, but had to drop a few issues due to space limitation. For example, one omission concerns so-called ABoxes which are frequently used to describe states of the world. ABoxes are not an extension of the concept language but rather situated "outside of it", similar to TBoxes (nevertheless, ABoxes are closely related to nominals, though much weaker). It seems that there exists no natural description logic with concrete domains for which reasoning with ABoxes is of a different complexity than reasoning without ABoxes. Some results on the combination of ABoxes and concrete domains can be found in, e.g., [Haarslev *et al.* 2001; Lutz 2002d; 2002b].

We should like to note that the research on description logics with concrete domains has already led to first reasoning systems that are equipped with concrete domains: the RACER system offers a concrete domain based on linear equations and inequalities resembling the concrete domain Q discussed in Section 4.1 [Haarslev & Möller 2002]. Moreover, there exist plans to extend the FaCT system [Horrocks 1998] with concrete domains. Since both RACER and FaCT provide for general TBoxes, they only offer the path-free variant of the concrete domain constructor discussed in Section 3.2. Serious implementations of description logics that provide for the full constructor remain yet to be seen.

BIBLIOGRAPHY

[Allen 1983] J. Allen. Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26(11), 1983.

[Areces *et al.* 2002] C. Areces, I. Horrocks, C. Lutz, and U. Sattler. Keys, nominals, and concrete domains. LTCS-Report 02-04, Technical University Dresden, 2002. To appear.

[Areces & de Rijke 2001] C. Areces and M. de Rijke. From description logics to hybrid logics, and back. In *Advances in Modal Logics Volume 3.* CSLI Publications, Stanford, CA, USA, 2001.

[Areces & Lutz 2002] C. Areces and C. Lutz. Concrete domains and nominals united. In *Proceedings of the fourth Workshop on Hybrid Logics (HyLo'02)*, 2002.

[Artale & Franconi 1998] A. Artale and E. Franconi. A temporal description logic for reasoning about actions and plans. *Journal of Artificial Intelligence Research (JAIR)*, 9:463–506, 1998.

[Artale & Franconi 2001] A. Artale and E. Franconi. Temporal description logics. In *Handbook of Time and Temporal Reasoning in Artificial Intelligence.* MIT Press, 2001. To appear.

[Artale & Lutz 1999] A. Artale and C. Lutz. A correspondence between temporal description logics. In *Proceedings of the International Workshop on Description Logics (DL'99)*, number 22 in CEUR-WS (http://ceur-ws.org/), pages 145–149, 1999.

[Baader *et al.* 2002a] F. Baader, I. Horrocks, and U. Sattler. Description logics for the semantic web. *KI – Künstliche Intelligenz*, 3, 2002. To appear.

[Baader *et al.* 2002b] F. Baader, C. Lutz, H. Sturm, and F. Wolter. Fusions of description logics and abstract description systems. *Journal of Artificial Intelligence Research (JAIR)*, 16:1–58, 2002.

[Baader 1999] F. Baader. Logic-based knowledge representation. In *Artificial Intelligence Today, Recent Trends and Developments*, number 1600 in Lecture Notes in Computer Science, pages 13–41. Springer-Verlag, 1999.

[Baader & Hanschke 1991] F. Baader and P. Hanschke. A scheme for integrating concrete domains into concept languages. In *Proceedings of the Twelfth International Joint Conference on Artificial Intelligence (IJCAI-91)*, pages 452–457, Sydney, Australia, 1991.

[Baader & Hanschke 1992] F. Baader and P. Hanschke. Extensions of concept languages for a mechanical engineering application. In *Proceedings of the 16th German AI-Conference (GWAI-92)*, vol. 671 of *Lecture Notes in Computer Science*, pages 132–143. Springer-Verlag, 1992.

[Baader & Sattler 2002] F. Baader and U. Sattler. Description logics with aggregates and concrete domains. *Information Systems*, 2002. To appear.

[Bennett 1997] B. Bennett. Modal logics for qualitative spatial reasoning. *Journal of the Interest Group in Pure and Applied Logic*, 4(1), 1997.

[Berners-Lee *et al.* 2001] T. Berners-Lee, J. Hendler, and O. Lassila. The semantic web. *Scientific American*, 284(5):34–43, 2001.

[Borgida 1996] A. Borgida. On the relative expressiveness of description logics and predicate logics. *Artificial Intelligence*, 82(1 - 2):353–367, 1996.

[Borgida & Weddell 1997] A. Borgida and G. E. Weddell. Adding uniqueness constraints to description logics (preliminary report). In *Proceedings of the 5th International Conference on Deductive and Object-Oriented Databases (DOOD97)*, vol. 1341 of *LNCS*, pages 85–102. Springer, 1997.

[Brachman *et al.* 1991] R. J. Brachman, D. L. McGuinness, P. F. Patel-Schneider, L. A. Resnick, and A. Borgida. Living with classic: When and how to use a KL-ONE-like language. In J. F. Sowa, editor, *Principles of Semantic Networks – Explorations in the Representation of Knowledge*, chapter 14, pages 401–456. Morgan Kaufmann, 1991.

[Calvanese *et al.* 1998] D. Calvanese, M. Lenzerini, and D. Nardi. Description logics for conceptual data modeling. In *Logics for Databases and Information Systems*, pages 229–263. Kluwer Academic Publisher, 1998.

[Calvanese *et al.* 2000] D. Calvanese, G. De Giacomo, and M. Lenzerini. Keys for free in description logics. In *Proceedings of the 2000 International Workshop in Description Logics (DL2000)*, number 33 in CEUR-WS (http://ceur-ws.org/), pages 79–88, 2000.

[Chen 1976] P. P.-S. Chen. The entity-relationship model-toward a unified view of data. *ACM Transactions on Database Systems*, 1(1):9–36, 1976.

[Edelmann & Owsnicki 1986] J. Edelmann and B. Owsnicki. Data models in knowledge representation systems: A case study. In *Proceedings of the Tenth German Workshop on Artificial Intelligence (GWAI'86) and the Second Austrian Symposium on Artificial Intelligence (ÖGAI'86)*, vol. 124 of *Informatik-Fachberichte*, pages 69–74. Springer Verlag, 1986.

[Fensel *et al.* 2000] D. Fensel, I. Horrocks, F. van Harmelen, S. Decker, M. Erdmann, and M. Klein. OIL in a nutshell. In *Proceedings of the European Knowledge Acquisition Conference (EKAW 2000)*, vol. 1937 of *Lecture Notes In Artificial Intelligence*. Springer-Verlag, 2000.

[Franconi & Ng 2000] E. Franconi and G. Ng. The i.com tool for intelligent conceptual modeling. In *Proceedings of the Seventh International Workshop on Knowledge Representation Meets Databases (KRDB2000)*, number 29 in CEUR-WS (http://ceur-ws.org/), pages 45–53, 2000.

[Giacomo & Lenzerini 1994] G. D. Giacomo and M. Lenzerini. Boosting the correspondence between description logics and propositional dynamic logics. In *Proceedings of the Twelfth National Conference on Artificial Intelligence (AAAI'94). Volume 1*, pages 205–212. AAAI Press, 1994.

[Grigorev 1988] D. Y. Grigorev. The complexity of deciding Tarski algebra. *Journal of Symbolic Computation*, 5(1,2):65–108, 1988.

[Haarslev *et al.* 1999] V. Haarslev, C. Lutz, and R. Möller. A description logic with concrete domains and role-forming predicates. *Journal of Logic and Computation*, 9(3):351–384, 1999.

[Haarslev *et al.* 2001] 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 First International Joint Conference on Automated Reasoning IJCAR'01*, number 2083 in Lecture Notes in Artifical Intelligence, pages 29–44. Springer-Verlag, 2001.

[Haarslev & Möller 2002] V. Haarslev and R. Möller. Practical reasoning in racer with a concrete domain for linear inequations. In *Proceedings of the International Workshop in Description Logics 2002 (DL2002)*, number 53 in CEUR-WS (http://ceur-ws.org/), 2002.

[Hanschke 1992] P. Hanschke. Specifying role interaction in concept languages. In W. Nebel, Bernhard; Rich, Charles; Swartout, editor, *Proceedings of the Third International Conference on Principles of Knowledge Representation and Reasoning (KR'92)*, pages 318–329. Morgan Kaufmann, 1992.

[Horrocks *et al.* 2000] I. Horrocks, U. Sattler, and S. Tobies. Practical reasoning for very expressive description logics. *Logic Journal of the IGPL*, 8(3):239–264, 2000.

[Horrocks 1998] I. Horrocks. Using an expressive description logic: Fact or fiction? In *Proceedings of the Sixth International Conference on the Principles of Knowledge Representation and Reasoning (KR98)*, pages 636–647, 1998.

[Horrocks & Patel-Schneider 2001] I. Horrocks and P. Patel-Schneider. The generation of DAML+OIL. In *Proceedings of the International Workshop in Description Logics 2001 (DL2001)*, number 49 in CEUR-WS (http://ceur-ws.org/), pages 30–35, 2001.

[Horrocks & Sattler 1999] I. Horrocks and U. Sattler. A description logic with transitive and inverse roles and role hierarchies. *Journal of Logic and Computation*, 9(3), 1999.

[Horrocks & Sattler 2001] I. Horrocks and U. Sattler. Ontology reasoning in the $\mathcal{SHOQ}(D)$ description logic. In B. Nebel, editor, *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence (IJCAI'01)*, pages 199–204. Morgan-Kaufmann, 2001.

[Khizder *et al.* 2001] V. L. Khizder, D. Toman, and G. E. Weddell. On decidability and complexity of description logics with uniqueness constraints. In *Proceedings of the 8th International Conference on Database Theory (ICDT2001)*, vol. 1973 of *LNCS*, pages 54–67. Springer, 2001.

[Kutz *et al.* 2001] O. Kutz, F. Wolter, and M. Zakharyaschev. A note on concepts and distances. In *Proceedings of the 2001 International Workshop in Description Logics (DL'01)*, number 49 in CEUR-WS (http://ceur-ws.org/), pages 113–121, 2001.

[Kutz *et al.* 2002] O. Kutz, F. Wolter, and M. Zakharyaschev. Connecting abstract description systems. In *Proceedings of the Eighth International Conference on Principles of Knowledge Representation and Reasoning (KR2002)*, pages 215–226. Morgan Kaufman, 2002.

[Lutz 1999] C. Lutz. Complexity of terminological reasoning revisited. In *Proceedings of the 6th International Conference on Logic for Programming and Automated Reasoning (LPAR'99)*, number 1705 in Lecture Notes in Artificial Intelligence, pages 181–200. Springer-Verlag, 1999.

[Lutz 2001a] C. Lutz. Interval-based temporal reasoning with general TBoxes. In B. Nebel, editor, *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence (IJCAI'01)*, pages 89–94. Morgan-Kaufmann, 2001.

[Lutz 2001b] C. Lutz. NExpTime-complete description logics with concrete domains. In *Proceedings of the First International Joint Conference on Automated Reasoning (IJCAR'01)*, number 2083 in Lecture Notes in Artifical Intelligence, pages 45–60. Springer-Verlag, 2001.

[Lutz 2002a] C. Lutz. Adding numbers to the $\mathcal{SHIQ}$ description logic—First results. In *Proceedings of the Eighth International Conference on Principles of Knowledge Representation and Reasoning (KR2002)*, pages 191–202. Morgan Kaufman, 2002.

[Lutz 2002b] C. Lutz. *The Complexity of Reasoning with Concrete Domains*. PhD thesis, LuFG Theoretical Computer Science, RWTH Aachen, Germany, 2002.

[Lutz 2002c] C. Lutz. NExpTime-complete description logics with concrete domains. *ACM Transactions on Computational Logic*, 2002. to appear.

[Lutz 2002d] C. Lutz. PSpace reasoning with the description logic $\mathcal{ALCF}(\mathcal{D})$. *Logic Journal of the IGPL*, 10(5):535–568, 2002.

[Lutz 2002e] C. Lutz. Reasoning about entity relationship diagrams with complex attribute dependencies. In *Proceedings of the International Workshop in Description Logics 2002 (DL2002)*, number 53 in CEUR-WS (http://ceur-ws.org/), pages 185–194, 2002.

[Lutz & Möller 1997] C. Lutz and R. Möller. Defined topological relations in description logics. In *Proceedings of the International Workshop on Description Logics (DL'97)*, pages 15–19, Gif sur Yvette (Paris), France, 1997. Université Paris-Sud, Centre d'Orsay.

[Mayr & Meyer 1982] E. W. Mayr and A. R. Meyer. The complexity of the word problem for commutative semigroups and polynomial ideals. *Advanced Mathematics*, 46:305–329, 1982.

[Molitor 2000] R. Molitor. *Unterstützung der Modellierung verfahrenstechnischer Prozesse durch Nicht-Standardinferenzen in Beschreibungslogiken*. PhD thesis, LuFG Theoretical Computer Science, RWTH Aachen, Germany, 2000.

[Nebel 1990] B. Nebel. Terminological reasoning is inherently intractable. *Artificial Intelligence*, 43:235–249, 1990.

[Pan & Horrocks 2002] J. Z. Pan and I. Horrocks. Reasoning in the $\mathcal{SHOQ}(\mathcal{D}_n)$ description logic. In *Proceedings of the International Workshop in Description Logics 2002 (DL2002)*, number 53 in CEUR-WS (http://ceur-ws.org/), pages 53–62, 2002.

[Randell *et al.* 1992] D. A. Randell, Z. Cui, and A. G. Cohn. A spatial logic based on regions and connection. In *Proceedings of the Third International Conference on Principles of Knowledge Representation and Reasoning (KR'92)*, pages 165–176. Morgan Kaufman, 1992.

[Renz & Nebel 1999] J. Renz and B. Nebel. On the complexity of qualitative spatial reasoning: A maximal tractable fragment of the region connection calculus. *Artificial Intelligence*, 108(1–2):69–123, 1999.

[Sattler 1998] U. Sattler. *Terminological knowledge representation systems in a process engineering application*. PhD thesis, LuFG Theoretical Computer Science, RWTH-Aachen, 1998.

[Schild 1991] K. D. Schild. A correspondence theory for terminological logics: Preliminary report. In *Proceedings of the Twelfth International Joint Conference on Artificial Intelligence (IJCAI-91)*, pages 466–471. Morgan Kaufmann, 1991.

[Schild 1993] K. D. Schild. Combining terminological logics with tense logic. In *Progress in Artificial Intelligence – 6th Portuguese Conference on Artificial Intelligence, EPIA'93*, vol. 727 of *Lecture Notes in Artificial Intelligence*, pages 105–120. Springer-Verlag, 1993.

[Schmidt-Schauß & Smolka 1991] M. Schmidt-Schauß and G. Smolka. Attributive concept descriptions with complements. *Artificial Intelligence*, 48(1):1–26, 1991.

[Schrijver 1986] A. Schrijver. *Theory of Linear and Integer Programming*. Wiley, Chichester, UK, 1986.

[Stock 1997] O. Stock, editor. *Spatial and Temporal Reasoning*. Kluwer Academic Publishers, Dordrecht, Holland, 1997.

[Teorey 1990] T. J. Teorey. *Database Modeling and Design - the Entity-Relationship Approach*. Morgan Kaufmann, 1990.

[van Benthem 1983] J. F. A. K. van Benthem. *Modal Logic and Classical Logic*. Bibliopolis, Naples, Italy, 1983.

[Wolter & Zakharyaschev 1999] F. Wolter and M. Zakharyaschev. Temporalizing description logic. In *Frontiers of Combining Systems*, pages 379 – 402. Studies Press/Wiley, 1999.