Chapter 1

NExpTime-complete Description Logics with Concrete Domains

Carsten Lutz

Abstract. Description Logics (DLs) incorporating concrete domains are useful formalisms for integrated reasoning about abstract and concrete knowledge. In this paper, we consider several extensions of $\mathcal{ALC}(\mathcal{D})$, which is the basic DL with concrete domains. We show that, although reasoning with $\mathcal{ALC}(\mathcal{D})$ is PSpace-complete, even "harmless-looking" extensions of this logic make reasoning NExpTime-complete.

1 Introduction

Description Logics (DLs) are a family of logical formalisms well-suited for the representation of and reasoning about conceptual knowledge on an abstract logical level. Many DLs are extensions of the basic propositionally complete Description Logic $\mathcal{ALC}$ which is a notational variant of the basic multi-modal logic $\mathbf{K}_m$. To see how knowledge is represented using $\mathcal{ALC}$, let us consider an example. The $\mathcal{ALC}$ concept

$$\mathit{Process} \sqcap \forall \mathit{workpiece}.\mathit{Metal} \sqcap \exists \mathit{workpiece}.\mathit{Large}$$

describes a manufacturing process which involves (i) only workpieces made from metal and (ii) at least one large workpiece. A DL concept corresponds to a formula in modal logics. In this example, *Process*, *Metal*, and *Large* are concept names (propositional variables) and *workpiece* is a role (an accessibility relation).

However, for many knowledge representation applications, it is essential to integrate the abstract logical knowledge with knowledge of a more concrete nature. For example, in the description of the manufacturing process, it may be important to be more precise about the meaning of "large". The standard technique for extending Description Logics in order to allow for the representation of concrete knowledge is to use so-called concrete domains which have been introduced by Baader and Hanschke in [1]. Baader and Hanschke define the description logic $\mathcal{ALC}(\mathcal{D})$, i.e., the extension of

$\mathcal{ALC}$ by concrete domains. More precisely, $\mathcal{ALC}(\mathcal{D})$ is parameterized with a concrete domain $\mathcal{D}$, where $\mathcal{D}$ provides a set of predicates over a given domain like, e.g., the real numbers or the set of time intervals. The concrete domain predicates can then be used inside a concrete domain concept constructor. Using $\mathcal{ALC}(\mathcal{D})$, we can describe the manufacturing process from above as

$$\mathit{Process} \sqcap \forall \mathit{workpiece}.\mathit{Metal} \sqcap \exists \mathit{workpiece}\ \mathit{diameter}.{\geq}50\mathit{cm}.$$

In this example, *diameter* is a role and $\geq 50cm$ is a concrete domain predicate. The last conjunct of this concept uses the concrete domain constructor to state that the process involves at least one workpiece whose diameter is at least 50 centimeters.

In this paper, we are interested in the complexity of reasoning with DLs which provide concrete domains. In [8], we proved that reasoning with $\mathcal{ALC}(\mathcal{D})$ is PSPACE-complete provided that reasoning with the concrete domain $\mathcal{D}$ (i.e., testing the satisfiability of finite conjunctions of predicates from $\mathcal{D}$) is in PSPACE. However, for many applications, the expressivity of $\mathcal{ALC}(\mathcal{D})$ is not sufficient and one wants to extend this logic by additional concept- and role-constructors. We investigate two such extensions and show that, in both extensions, reasoning becomes considerably harder. More precisely, we consider the extension of $\mathcal{ALC}(\mathcal{D})$ by acyclic TBoxes and inverse roles. Acyclic TBoxes can be thought of as acyclic macro definitions. For example, a process whose duration is at least one hour can be named *Long-Process*:

$$\mathit{Long\text{-}Process} \doteq \mathit{Process} \sqcap \exists \mathit{start}, \mathit{end}.{\geq}1h.$$

Inverse roles correspond to converse modalities and allow, e.g., to describe cylinders whose manufacturing takes a long time as

$$\mathit{Cylinder} \sqcap \exists \mathit{workpiece}^{-}.\mathit{Long\text{-}Process}$$

By introducing a NEXPTIME-complete variant of the Post Correspondence Problem [10, 5], we show that there exists a concrete domain $\mathcal{P}$ for which reasoning is in PTIME such that reasoning with each of the above two extensions of $\mathcal{ALC}(\mathcal{D})$ (parameterized with the concrete domain $\mathcal{P}$) is NEXPTIME-hard. This dramatic increase in complexity is rather surprising since, from a computational point of view, all of the proposed extensions look harmless. For example, in [7], we show that the extension of "many" PSPACE Description Logics by acyclic TBoxes does not increase the complexity of reasoning. Moreover, it is well-known that $\mathcal{ALC}$ extended by inverse roles is still in PSPACE (see, e.g., [12, 6]). As a corresponding upper bound, we show that, if reasoning with a concrete domain $\mathcal{D}$ is in NP, then reasoning with the DL $\mathcal{ALCI}(\mathcal{D})$ (i.e., the extension of $\mathcal{ALC}(\mathcal{D})$ with inverse roles) with acyclic TBoxes is in NEXPTIME. This paper is accompanied by a technical report which contains all proofs and technical details [9].

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

In this section, we formally introduce the description logic $\mathcal{ALCI}(\mathcal{D})$ with which we are concerned in the remainder of this paper.

Definition 1 (Concrete Domain). A *concrete domain* $\mathcal{D} = (\Delta_\mathcal{D}, \Phi_\mathcal{D})$ is given by a a set $\Delta_\mathcal{D}$ called the domain and a set of predicate names $\Phi_\mathcal{D}$. Each predicate name $P \in \Phi_\mathcal{D}$ is associated with an arity n and an n-ary predicate $P^\mathcal{D} \subseteq \Delta_\mathcal{D}^n$. A concrete domain $\mathcal{D}$ is called *admissible* iff (1) the set of its predicate names is closed under negation and contains a name $\top_\mathcal{D}$ for $\Delta_\mathcal{D}$ and (2) the satisfiability problem for finite conjunctions of predicates is decidable.

We will only admit concrete domains which are admissible. Based on concrete domains, we introduce the syntax of $\mathcal{ALCI}(\mathcal{D})$.

Definition 2 (Syntax). Let N_C, N_R, and N_{cF} be mutually disjoint sets of *concept names*, *role names*, and *concrete feature names*, respectively, and let N_{aF} be a subset of N_R. Elements of N_{aF} are called *abstract features*. The set of $\mathcal{ALCI}(\mathcal{D})$ roles $\overline{N_R}$ is $N_R \cup \{R^- \mid R \in N_R\}$. An expression $f_1 \cdots f_n g$, where $f_1, \ldots, f_n \in N_{aF}$ and $g \in N_{cF}$, is called a *path*.[1] The set of $\mathcal{ALCI}(\mathcal{D})$ concepts is the smallest set such that (1) every concept name is a concept, and (2) if C and D are concepts, R is a role, g is a concrete feature, $P \in \Phi$ is a predicate name with arity n, and $u_1, \ldots, u_n$ are paths, then the following expressions are also concepts:

$$\neg C,\ C \sqcap D,\ C \sqcup D,\ \exists R.C,\ \forall R.C,\ \exists u_1, \ldots, u_n.P,\ \text{and}\ g{\uparrow}.$$

An $\mathcal{ALCI}(\mathcal{D})$ concept which uses only roles from N_R is called an $\mathcal{ALC}(\mathcal{D})$ concept. In the following, we denote concepts with C and D, roles with R, abstract features with f, concrete features with g, and predicates with P.

Definition 3 (TBoxes). Let A be a concept name and C be a concept. Then $A \doteq C$ is a *concept definition*. Let $\mathcal{T}$ be a finite set of concept definitions. A concept name A *directly uses* a concept name B in $\mathcal{T}$ if there is a concept definition $A \doteq C$ in $\mathcal{T}$ such that B appears in C. Let *uses* be the transitive closure of "directly uses". $\mathcal{T}$ is called *acyclic* if there is no concept name A such that A uses itself in $\mathcal{T}$. If $\mathcal{T}$ is acyclic, and the left-hand sides of all concept definitions in $\mathcal{T}$ are unique, then $\mathcal{T}$ is called a *TBox*.

As usual, a set-theoretic semantics for concepts and TBoxes is given.

Definition 4 (Semantics). An *interpretation* is a pair $(\Delta_\mathcal{I}, \cdot^\mathcal{I})$, where $\Delta_\mathcal{I}$ is a set called the *domain* and $\cdot^\mathcal{I}$ the *interpretation function*. The interpretation function maps each concept name C to a subset $C^\mathcal{I}$ of $\Delta_\mathcal{I}$, each

[1] A concrete feature is a path of length 1.

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 $u^{\mathcal{I}}(a)$ is defined as $f_1^{\mathcal{I}} \circ \cdots \circ f_n^{\mathcal{I}} \circ g^{\mathcal{I}}$, where $\circ$ denotes function composition and $f_1 \circ f_2(a) = f_2(f_1(a))$ for f_1 and f_2 functions. The interpretation function is extended to arbitrary roles and concepts as follows:

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

Let C be a concept and $\mathcal{T}$ be a TBox. If $C^{\mathcal{I}} \neq \emptyset$, then $\mathcal{I}$ is called a *model* for C. If $A^{\mathcal{I}} = C^{\mathcal{I}}$ for all $A \doteq C \in \mathcal{T}$, then $\mathcal{I}$ is called a *model* for $\mathcal{T}$.

We call elements of $\Delta_{\mathcal{I}}$ *abstract objects* and elements of $\Delta_{\mathcal{D}}$ *concrete objects*. In their original $\mathcal{ALC}(\mathcal{D})$ definition, Baader and Hanschke define only one type of features which are interpreted as partial functions from $\Delta_{\mathcal{I}}$ to $\Delta_{\mathcal{I}} \cup \Delta_{\mathcal{P}}$ [1]. We chose a different variant since separating concrete and abstract features allows a clearer algorithmic treatment and the loss in expressivity is only marginal: it cannot be expressed that a single feature f relates a given object to an abstract *or* a concrete object. However, we never encountered the need to do this when representing knowledge with $\mathcal{ALC}(\mathcal{D})$ and its relatives.

Definition 5 (Inference Problems). Let C and D be concepts. *C subsumes D w.r.t. a TBox $\mathcal{T}$* (written $D \sqsubseteq_{\mathcal{T}} C$) iff $D^{\mathcal{I}} \subseteq C^{\mathcal{I}}$ for all models $\mathcal{I}$ of $\mathcal{T}$. *C is satisfiable w.r.t. a TBox $\mathcal{T}$* iff there exists a model of both $\mathcal{T}$ and C. Both inferences are also considered without reference to TBoxes: *C subsumes D* iff C subsumes D w.r.t. the empty TBox. *C is satisfiable* iff it is satisfiable w.r.t. the empty TBox.

It is well-known that (un)satisfiability and subsumption can be mutually reduced to each other, i.e., $C \sqsubseteq_{\mathcal{T}} D$ iff $C \sqcap \neg D$ is unsatisfiable w.r.t. $\mathcal{T}$ and C is satisfiable w.r.t. $\mathcal{T}$ iff we do not have $C \sqsubseteq_{\mathcal{T}} A \sqcap \neg A$ for some $A \in N_C$. This connection allows us to concentrate on satisfiability for the remainder of this paper.

3 Complexity Results

The main contribution of this paper are NExpTime-completeness results for the logics introduced in the previous section. However, since deciding

satisfiability of $\mathcal{ALC}(\mathcal{D})$ or $\mathcal{ALCI}(\mathcal{D})$ concepts (possibly w.r.t. TBoxes) involves reasoning with the concrete domain $\mathcal{D}$, the complexity of reasoning with these DLs depends on the complexity of reasoning with the concrete domain $\mathcal{D}$. As the definition of admissible concrete domains suggests, the task performed by the concrete domain reasoner is usually to decide the satisfiability of finite conjunctions of concrete domain predicates [2, 8]. The complexity of this inference will thus be important for establishing complexity results for $\mathcal{ALC}(\mathcal{D})$ and $\mathcal{ALCI}(\mathcal{D})$.

3.1 $2^n + 1$-PCPs and the Concrete Domain $\mathcal{P}$

The lower complexity bounds are established by a reduction of the $2^n + 1$-PCP, a NExpTime-complete variant of the well-known Post Correspondence Problem [10] which we introduce in the following. We also introduce a concrete domain $\mathcal{P}$ that can be used to reduce the 2^n+1-PCP to the satisfiability of $\mathcal{ALC}(\mathcal{D})$ concepts w.r.t. TBoxes and $\mathcal{ALCI}(\mathcal{D})$ concepts, and determine the complexity of reasoning with $\mathcal{P}$.

Definition 6 (PCP). A *Post Correspondence Problem (PCP)* P is given by a finite, non-empty list $(\ell_1, r_1), \ldots, (\ell_k, r_k)$ of pairs of non-empty words over some infinite alphabet Σ. A sequence of integers $i_1, \ldots, i_m$, with $m \geq 1$, is called a *solution* for P iff $\ell_{i_1} \cdots \ell_{i_m} = r_{i_1} \cdots r_{i_m}$. Let $f(n)$ be a mapping from $\mathbb{N}$ to $\mathbb{N}$ and let $|P|$ denote the sum of the lengths of all words in the PCP $P = (\ell_1, r_1), \ldots, (\ell_k, r_k)$, i.e.,

$$|P| = \sum_{1 \leq i \leq k} |\ell_i| + |r_i|.$$

A solution $i_1, \ldots, i_m$ is called an $f(n)$-*solution* iff $m \leq f(|P|)$. With $f(n)$-*PCP*, we denote the version of the PCP that admits only $f(n)$-solutions.

Hence, a $2^n + 1$-PCP P admits only solutions $i_1, \ldots, i_m$ with $m \leq 2^{|P|} + 1$. By modifying the proof used by Hopcroft and Ullman to show that the general PCP is undecidable [5], we prove the following theorem.

Theorem 7. *It is* NExpTime*-complete to decide whether a* $2^n + 1$*-PCP has a solution.*

In order to reduce the $2^n + 1$-PCP to concept satisfiability, we introduce an appropriate concrete domain $\mathcal{P}$.

Definition 8 (Concrete Domain $\mathcal{P}$). Let Σ be an alphabet. The concrete domain $\mathcal{P}$ is defined by setting $\Delta_{\mathcal{P}} := \Sigma^*$ and defining $\Phi_{\mathcal{P}}$ as the smallest set containing the following predicates:

- unary predicates *word* and *nword* with $word^{\mathcal{P}} = \Delta_{\mathcal{P}}$ and $nword^{\mathcal{P}} = \emptyset$,
- unary predicates $=_\epsilon$ and $\neq_\epsilon$ with $=_\epsilon^{\mathcal{P}} = \{\epsilon\}$ and $\neq_\epsilon^{\mathcal{P}} = \Sigma^+$,

- a binary equality predicate $=$ and a binary inequality predicate $\neq$, and
- for each $w \in \Sigma^+$, two binary predicate $conc_w$ and $nconc_w$ with

$$conc_w^{\mathcal{P}} = \{(u, v) \mid v = uw\} \text{ and } nconc_w^{\mathcal{P}} = \{(u, v) \mid v \neq uw\}.$$

To show that $\mathcal{P}$ is admissible, we need to prove that the satisfiability of finite conjunctions of predicates from $\mathcal{P}$ is decidable. As an example for such a conjunction, consider $word(x) \wedge conc_w(x, y) \wedge =(x, y)$. We prove decidability by devising an algorithm based on repeated normalization combined with tests for obvious inconsistencies. The algorithm does also provide an upper bound for the complexity of reasoning with the concrete domain $\mathcal{P}$.

Proposition 9. *It is decidable in deterministic polynomial time whether a finite conjunction of predicates from $\mathcal{P}$ has a solution.*

We claim that the concrete domain $\mathcal{P}$ is rather natural since, instead of words and predicates of words, one could use natural numbers and simple operations on natural numbers: Words over an alphabet Σ can be interpreted as numbers written at base $|\Sigma| + 1$ (assuming that the empty word represents 0); the concatenation of two words v and w can then be expressed as $vw = v * (|\Sigma| + 1)^{|w|} + w$, where $|w|$ denotes the length of the word w [3]. Hence, a concrete domain which provides the natural numbers, (in)equality, (in)equality to zero, addition, and multiplication is also appropriate for the reductions.

3.2 Lower and Upper Bounds

As lower bounds, we show that (i) satisfiability of $\mathcal{ALC}(\mathcal{P})$ concepts w.r.t. TBoxes and (ii) satisfiability of $\mathcal{ALCI}(\mathcal{P})$ concepts without reference to TBoxes are NExpTime-hard. As a corresponding upper bound, it can be shown that satisfiability of $\mathcal{ALCI}(\mathcal{D})$ concepts w.r.t. TBoxes is in NExpTime if reasoning with the concrete domain $\mathcal{D}$ is in NP. Due to space limitations, however, we only sketch the proof of the upper bound.

We start with proving the lower bound for the satisfiability of $\mathcal{ALC}(\mathcal{D})$ concepts w.r.t. TBoxes. Given a $2^n + 1$-PCP $P = (\ell_1, r_1), \ldots, (\ell_k, r_k)$, we define a TBox $\mathcal{T}[P]$ of size polynomial in $|P|$ and a concept (name) C such that C is satisfiable w.r.t. $\mathcal{T}[P]$ iff P has a solution. The reduction TBox can be found in Figure 1.1, where ℓ, r, x, and y are abstract features. The first equality in the figure is not meant as a concept definition but as an abbreviation: Replace every occurrence of $Ch[u_1, u_2, u_3, u_4]$ in the lower three concept definitions by the right-hand side of the first equation substituting $u_1, \ldots, u_4$ appropriately. Moreover, if f is a feature, we use $\exists f^n.C$ to denote the n-fold nesting $\exists f. \cdots \exists f.C$. In the following, we give an informal explanation of the strategy underlying the reduction.

$$
\begin{aligned}
Ch[u_1,u_2,u_3,u_4] &= (\exists(u_1,u_2).= \sqcap \exists(u_3,u_4).=) \\
&\quad \sqcup \bigsqcup_{(\ell_i,r_i) \text{ in } P} (\exists(u_1,u_2).conc_{\ell_i} \sqcap \exists(u_3,u_4).conc_{r_i}) \\
\\
C_1 &\doteq \exists\ell.C_2 \sqcap \exists r.C_2 \\
&\quad \sqcap Ch[\ell r^{n-1} g_\ell, r\ell^{n-1} g_\ell, \ell r^{n-1} g_r, r\ell^{n-1} g_r] \\
&\quad \vdots \\
C_{n-2} &\doteq \exists\ell.C_{n-1} \sqcap \exists r.C_{n-1} \\
&\quad \sqcap Ch[\ell r g_\ell, r\ell g_\ell, \ell r g_r, r\ell g_r] \\
C_{n-1} &\doteq Ch[\ell g_\ell, r g_\ell, \ell g_r, r g_r] \\
\\
C &\doteq C_1 \\
&\quad \sqcap \exists \ell^n g_\ell. =_\epsilon \sqcap \exists \ell^n g_r. =_\epsilon \\
&\quad \sqcap \exists r^n y. \exists g_\ell, g_r. = \sqcap \exists r^n y g_\ell. \neq_\epsilon \\
&\quad \sqcap Ch[r^n g_\ell, r^n x g_\ell, r^n g_r, r^n x g_r] \\
&\quad \sqcap Ch[r^n x g_\ell, r^n y g_\ell, r^n x g_r, r^n y g_r]
\end{aligned}
$$

Figure 1.1: The $\mathcal{ALC}(\mathcal{P})$ reduction TBox $\mathcal{T}[P]$ ($n = |P|$).

The general idea is to define $\mathcal{T}[P]$ such that models of C and $\mathcal{T}[P]$ have the form of a binary tree of depth $|P|$ whose edges are connected by two "chains" of $conc_w$ predicates (see Figure 1.2 for an example model). Pairs of corresponding objects (x_i, y_i) on the chains represent "partial solutions" of the PCP P. More precisely, the first line of the definitions of the $C_1, \ldots, C_{n-1}$ concepts ensure that models $\mathcal{I}$ of C and $\mathcal{T}[P]$ have the form of a binary tree of depth n (with $n = |P|$) whose left edges are labeled with the abstract feature ℓ and whose right edges are labeled with the abstract feature r The nodes in the tree must not necessarily be distinct. Let the abstract objects $a_0, \ldots a_{2^n-1}$ be the leaves of the tree. A careful analysis of the second line of the definitions of the $C_1, \ldots, C_{n-1}$ concepts and the definition of the Ch concept reveals that

1. every a_i ($0 \leq i < 2^n$) has a filler x_i for the concrete feature g_ℓ and a filler y_i for the concrete feature g_r, and,

2. for $0 \leq i < 2^n - 1$, either $x_i = x_{i+1}$ and $y_i = y_{i+1}$, or there exists a $j \in \{1, \ldots, k\}$ such that $(x_i, x_{i+1}) \in conc^{\mathcal{P}}_{\ell_j}$ and $(y_i, y_{i+1}) \in conc^{\mathcal{P}}_{r_j}$.

Since we must consider solutions of a length up to $2^n + 1$, the 2^n objects on the fringe of the tree are not sufficient and we need to "add" two more objects a_{2^n} and a_{2^n+1} which behave analogously to the objects $a_0, \ldots a_{2^n-1}$, i.e., have associated concrete objects x_{2^n}, y_{2^n} and x_{2^n+1}, y_{2^n+1}, respectively.

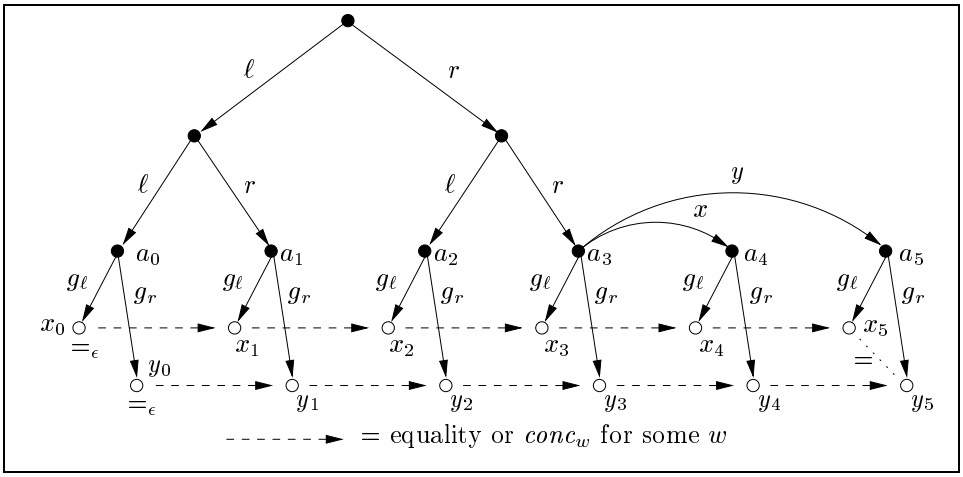

Figure 1.2: An example model of C for $n = 2$.

This is done by the last two lines of the definition of C. Finally, the second line of the definition of C ensures that $x_0 = y_0 = \epsilon$ and the third line ensures that $x_{2^n+1} = y_{2^n+1} \neq \epsilon$. Hence, (x_{2^n+1}, y_{2^n+1}) is a solution of the PCP P. Using the above considerations, the correctness of the reduction can be formally proved and thus we obtain the following theorem.

Theorem 10. *There exists an admissible concrete domain $\mathcal{D}$, for which satisfiability is in* PTime, *such that satisfiability and subsumption of $\mathcal{ALC}(\mathcal{D})$ concepts w.r.t TBoxes are* NExpTime*-hard.*

For the $\mathcal{ALCI}(\mathcal{P})$ reduction, we define a concept $C[P]$ which is satisfiable iff the PCP P has a solution. The $\mathcal{ALCI}(\mathcal{P})$ reduction is similar to the $\mathcal{ALC}(\mathcal{P})$ reduction since (i) we use two predicate chains to represent partial solutions, and (ii) we use a tree of abstract objects to generate the chains. However, in the $\mathcal{ALCI}(\mathcal{P})$ reduction, the chains are generated in a different way. They do not connect the leaves of the tree but are "laid around" all nodes in the tree as indicated in Figure 1.4. The reduction concept can be found in Figure 1.3, where $h_\ell, h_r, x_\ell, x_r, y_\ell, y_r, z_\ell$, and z_r are concrete features. All equalities in this Figure are intended as abbreviations and not as concept definitions. We did not repeat the definition of *Ch* since it can be found in Figure 1.1.

Again, we only give an informal discussion of the reduction. Due to the first line in the definition of $C[P]$ and the $\exists f^-$ quantifiers in the definition of X, models of $C[P]$ have the form of a tree of depth $|P| - 1$ in which all edges are labeled with f^-. This edge labelling scheme is possible since the inverse of an abstract feature is not a feature. The existence of the predicate chains is ensured by the definition of X and the second line in the definition of $C[P]$. Figure 1.5 shows a detailed clipping from a model of $C[P]$ (in

$$
\begin{aligned}
X = {} & \exists f^-(Ch[fg_\ell, g_\ell, fg_r, g_r] \sqcap \exists(h_\ell, fp_\ell).= \sqcap \exists(h_r, fp_r).=) \\
& \sqcap \exists f^-(Ch[fp_\ell, g_\ell, fp_r, g_r] \sqcap \exists(h_\ell, fh_\ell).= \sqcap \exists(h_r, fh_r).=)
\end{aligned}
$$

$$
\begin{aligned}
C[P] = {} & X \sqcap \forall f^-.X \sqcap \cdots \sqcap \forall (f^-)^{n-1}.X \\
& \sqcap \forall (f^-)^n.(\exists(g_\ell, h_\ell).= \sqcap \exists(g_r, h_r).=) \\
& \sqcap Ch[h_\ell, x_\ell, h_r, x_r] \sqcap Ch[x_\ell, y_\ell, x_r, y_r] \sqcap Ch[y_\ell, z_\ell, y_r, z_r] \\
& \sqcap \exists g_\ell, =_\epsilon \sqcap \exists g_r, =_\epsilon \sqcap \exists z_\ell, z_r.= \sqcap \exists z_\ell.\neq_\epsilon
\end{aligned}
$$

Figure 1.3: The $\mathcal{ALCI}(\mathcal{P})$ reduction concept $C[P]$ ($n = |P| - 1$).

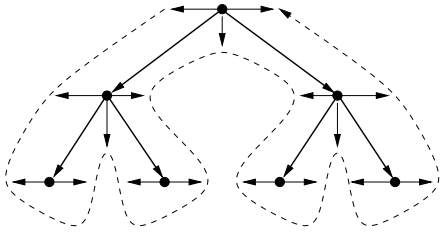

Figure 1.4: Predicate chains in models of $C[P]$.

fact, Figure 1.5 shows a model of the concept X). The second line of $C[P]$ establishes the edges "leading around" the fringe nodes.

The length of the two predicate chains is twice the length of the number of edges in the tree plus the number of fringe nodes, i.e., $2*(2^{|P|}-2)+2^{|P|-1}$. To eliminate the factor 2 and the summand $2^{|P|-1}$, $C[P]$ is defined such that every edge in the predicate chains leading "up" in the tree and every edge "leading around" a fringe node is labeled with the equality predicate. To extend the chains to length $2^{|P|}+1$, we need to add three additional edges. This is reflected in the third line of the definition of $C[P]$. Finally, the last line in the definition of $C[P]$ ensures that the first concrete object on each chain represents the empty word and that the last pair of corresponding objects represents a (non-empty) solution for P.

Theorem 11. *There exists an admissible concrete domain $\mathcal{D}$, for which satisfiability is in* PTIME*, such that satisfiability and subsumption of $\mathcal{ALCI}(\mathcal{D})$ concepts are* NEXPTIME*-hard.*

The upper bound is proved by devising a tableau algorithm which is capable of deciding the satisfiability of $\mathcal{ALCI}(\mathcal{D})$ concepts w.r.t. TBoxes. Tableau algorithms decide the satisfiability of concepts C by trying to build a canonical model for C. To do this, completion rules are repeatedly applied to so-called completion systems until, finally, either a contradiction is found—

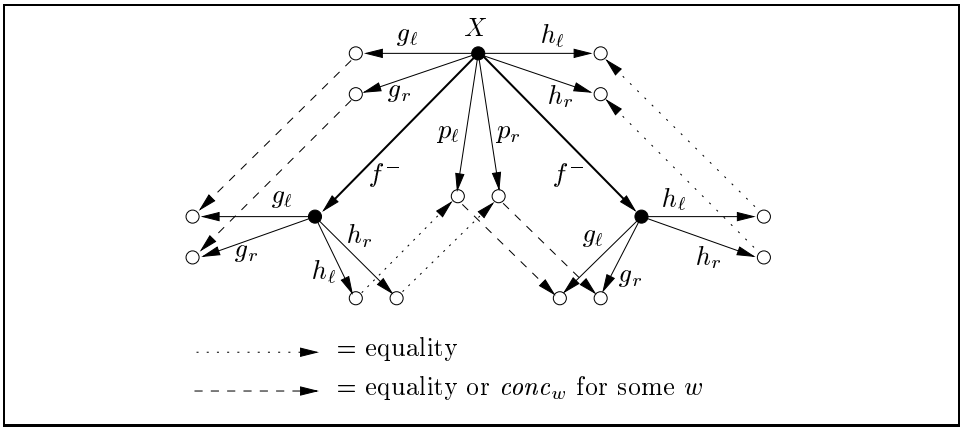

Figure 1.5: A clipping from a model of $C[P]$.

meaning that C does not have a model—or a completion system is obtained to which no completion rule is applicable and which represents a model for C. A tableau algorithm for deciding the satisfiability of $\mathcal{ALCI}(\mathcal{D})$ concepts without reference to TBoxes can be given by straightforwardly combining known tableau algorithms for the DLs $\mathcal{ALC}(\mathcal{D})$ and $\mathcal{ALCI}$. The algorithm is the modified to take into account TBoxes by using a technique introduced in [7]. A complexity analysis of the algorithm yields the following theorem.

Theorem 12. *If satisfiability of the concrete domain $\mathcal{D}$ is in* NP*, satisfiability of $\mathcal{ALCI}(\mathcal{D})$ concepts w.r.t. TBoxes can be decided in nondeterministic exponential time.*

4 Conclusion

As future work, we plan to extend the obtained logics by further constructors such as transitive roles [11] and qualifying number restrictions [4]. There are at least two ways to go: One could define extensions of $\mathcal{ALCF}(\mathcal{D})$ (see [8]) trying to obtain an expressive logic with concrete domains which is still in PSpace. The second approach is to define extensions of $\mathcal{ALCI}(\mathcal{D})$ which means that the obtained logics are at least NExpTime-hard and that feature (dis)agreements cannot be considered without loosing decidability (in [9], we prove that the DL $\mathcal{ALCIF}$ is undecidable).

Acknowledgments My thanks go to Franz Baader, Ulrike Sattler, and Stephan Tobies for inspiring discussions. The work in this paper was supported by the DFG Project BA1122/3-1 "Combinations of Modal and Description Logics".

Bibliography

[1] 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, August 24–30, 1991. Morgan Kaufmann Publ. Inc., San Mateo, CA, 1991.

[2] F. Baader and P. Hanschke. A scheme for integrating concrete domains into concept languages. DFKI Research Report RR-91-10, German Research Center for Artificial Intelligence, Kaiserslautern, 1991.

[3] F. Baader and P. Hanschke. Extensions of concept languages for a mechanical engineering application. In *Proceedings of the 16th German AI-Conference, GWAI-92*, volume 671 of *Lecture Notes in Computer Science*, pages 132–143, Bonn (Germany), 1993. Springer–Verlag.

[4] B. Hollunder and F. Baader. Qualifying number restrictions in concept languages. In *Proceedings of the Second International Conference KR'91*, pages 335–346, Cambridge, Mass., April 22–25, 1991. Morgan Kaufmann Publ. Inc., San Mateo, CA, 1991. DFKI Research Report RR-91-03, Kaiserslautern.

[5] J. E. Hopcroft and J. D. Ullman. *Introduction to Automata Theory, Languages and Computation*. Addison-Wesley, Reading, Mass., 1979.

[6] I. Horrocks, U. Sattler, and S. Tobies. Practical reasoning for expressive description logics. 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 161–180. Springer-Verlag, Sept. 1999.

[7] 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, Sept. 1999.

[8] C. Lutz. Reasoning with concrete domains. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence IJCAI-99*, Stockholm, Sweden, July 31 – August 6, 1999.

[9] C. Lutz. NExpTime-complete description logics with concrete domains. LTCS-Report 00-01, LuFG Theoretical Computer Science, RWTH Aachen, Germany, 2000. See http://www-lti.informatik.rwth-aachen.de/Forschung/Reports.html.

[10] E. M. Post. A variant of a recursively unsolvable problem. *Bull. Am. Math. Soc.*, 52:264–268, 1946.

[11] U. Sattler. A concept language extended with different kinds of transitive roles. In *20. Deutsche Jahrestagung für Künstliche Intelligenz*, number 1137 in Lecture Notes in Artificial Intelligence. Springer Verlag, 1996.

[12] E. Spaan. The complexity of propositional tense logics. In *Diamonds and Defaults*, pages 287–307. Kluwer Academic Publishers, 1993.