Dresden University of Technology
Institute for Theoretical Computer Science
Chair for Automata Theory

LTCS–Report

LTL over Description Logic Axioms

Franz Baader   Silvio Ghilardi   Carsten Lutz

LTCS-Report 08-01

Lehrstuhl für Automatentheorie
Institut für Theoretische Informatik
TU Dresden
http://lat.inf.tu-dresden.de

Nöthnitzer Str. 46
01187 Dresden
Germany

LTL over Description Logic Axioms

Franz Baader*
TU Dresden, Germany
baader@inf.tu-dresden.de

Silvio Ghilardi
Università degli Studi di Milano, Italy
ghilardi@dsi.unimi.it

Carsten Lutz
TU Dresden, Germany
lutz@inf.tu-dresden.de

April 10, 2008

Abstract

Most of the research on temporalized Description Logics (DLs) has concentrated on the case where temporal operators can occur within DL concept descriptions. In this setting, reasoning usually becomes quite hard if rigid roles, i.e., roles whose interpretation does not change over time, are available. In this paper, we consider the case where temporal operators are allowed to occur only in front of DL axioms (i.e., ABox assertions and general concept inclusion axioms), but not inside of concepts descriptions. As the temporal component, we use linear temporal logic (LTL) and in the DL component we consider the basic DL $\mathcal{ALC}$. We show that reasoning in the presence of rigid roles becomes considerably simpler in this setting.

1 Introduction

Description logics (DLs) [7] are a family of logic-based knowledge representation formalisms, which are employed in various application domains, such as natural language processing, configuration, databases, and bio-medical ontologies, but their most notable success so far is the adoption of the DL-based language OWL

*supported by NICTA, Canberra Research Lab.

[13] as standard ontology language for the semantic web. In many applications of (DLs), such as the use of DLs as ontology languages or conceptual modeling languages, being able to represent dynamic aspects of the application domain would be quite useful. This is, for instance, the case if one wants to use DLs as conceptual modeling languages for temporal databases [6]. Another example are medical ontologies, where the faithful representation of concepts would often require the description of temporal patterns. As a simple example, consider the concept "Concussion with no loss of consciousness," which is modeled as a primitive (i.e., not further defined) concept in the medical ontology SNOMED CT.[1] As argued in [21], a correct representation of this concept should actually say that, after the concussion, the patient remained conscious until the examination.

Since the expressiveness of pure DLs is not sufficient to describe such temporal patterns, a plethora of temporal extensions of DLs have been investigated in the literature.[2] These include approaches as diverse as the combination of DLs with Halpern and Shoham's logic of time intervals [20], formalisms inspired by action logics [3], the treatment of time points and intervals as a concrete domains [14], and the combination of standard DLs with standard (propositional) temporal logics into logics with a two-dimensional semantics, where one dimension is for time and the other for the DL domain [18, 23, 12]. In this paper, we follow the last approach, where we use the basic DL $\mathcal{ALC}$ [19] in the DL component and linear temporal logic (LTL) [16] (sometimes also called propositional temporal logic (PTL) [12]) in the temporal component. However, even after the DL and the temporal logic to be combined have been fixed, there remain several degrees of freedom when defining the resulting temporalized DL.

On the one hand, one must decide to which pieces of syntax temporal operators can be applied. Temporal operators may be allowed to occur within concept descriptions, as required by the above example of a concussion with no loss of consciousness, which could be defined using the until-operator U of LTL as follows:

$$\exists \mathsf{finding}.\mathsf{Concussion} \sqcap \mathsf{Conscious}\,\mathsf{U}\,\exists \mathsf{procedure}.\mathsf{Examination}. \tag{1}$$

Alternatively or in addition, temporal operators may be applied to TBox axioms (i.e., general concept inclusions, GCIs) and/or to ABox assertions. For example, the temporalized TBox axiom

$$\Diamond\Box(\mathsf{UScitizen} \sqsubseteq \exists \mathsf{insured\_by}.\mathsf{HealthInsurer})$$

says that there is a future time point from which on US citizens will always have health insurance, and the formula Ψ:

$$\begin{aligned}\Diamond\big((\exists \mathsf{finding}.\mathsf{Concussion})(\mathsf{BOB})\ \wedge \\ \mathsf{Conscious}(\mathsf{BOB})\mathsf{U}(\exists \mathsf{procedure}.\mathsf{Examination})(\mathsf{BOB})\big)\end{aligned} \tag{2}$$

[1] see http://www.ihtsdo.org/our-standards/

[2] For a more thorough survey of the literature on temporalized DLs, see the survey papers [4, 5, 15].

says that, sometime in the future, Bob will have a concussion with no loss of consciousness between the concussion and the examination.

On the other hand, one must decide whether one wants to have rigid concepts and/or roles, i.e., concepts/roles whose interpretation does not vary over time. For example, the concept Human and the role has_father should probably be rigid since a human being will stay a human being and have the same father over his/her life-time, whereas Conscious should be a flexible concept (i.e., not rigid) since someone that is conscious at the moment need not always by conscious. Similarly, insured_by should be modeled as a flexible role. Using a logic that cannot enforce rigidity of concepts/roles may result in unintended models, and thus prevent certain useful inferences to be drawn. For example, the concept description $\exists \mathsf{has\_father}.\mathsf{Human} \sqcap \Diamond (\forall \mathsf{has\_father}.\neg\mathsf{Human})$ is only unsatisfiable if both has_father and Human are rigid.

Related work

The combination of $\mathcal{ALC}$ with LTL was first considered by Schild [18] and, since then, has developed into a lively research area recently surveyed in [15]. In the temporalized DL proposed by Schild, temporal operators can be applied to concept descriptions, but not to TBox axioms (and ABoxes are not considered at all). Rigid concepts are definable in this logic, but rigid roles are not. Schild observes that his logic behaves similarly to the so-called *fusion* of $\mathcal{ALC}$ and LTL,[3] which shows that the interaction between the $\mathcal{ALC}$ component and the LTL component is limited. This observation forms the basis for Schild's proof that reasoning in his logic is ExpTime-complete.

The combination of (extensions of) $\mathcal{ALC}$ and LTL in which temporal operators are applied to concept descriptions, TBox axioms, and ABox assertions has been studied by Wolter, Zakharyaschev, and others (see, e.g., [23, 12]). In this more general setting, the interaction between the DL component and LTL is much stronger, and the complexity of reasoning is ExpSpace-complete. As in Schild's logic, rigid concepts can be defined, but rigid roles cannot. As also shown in [12], the addition of rigid roles causes undecidability. This already holds for concept satisfiability w.r.t. a global TBox (i.e., where the same TBox axioms must hold at all time points) and with only a single rigid role. Decidability can be regained by dropping TBoxes altogether, but the decision problem is still hard for non-elementary time [12].

Decidable combinations of DLs and temporal logics that allow for rigid roles can be obtained by restricting either the temporal or the DL component. In [2], temporal operators can be applied to concept descriptions, TBoxes are global, and there are no ABoxes. The reason for decidability (more precisely, 2-ExpTime-completeness) also in the presence of rigid role is that the only available temporal

[3]Note, however, that Schild's proof of this is incorrect; a correct proof can be found in [15].

operators are an undirected diamond expressing "at some time point" and an undirected box expressing "at all time points." Here, undirected means that these operators cannot discriminate between the past, the future, and the current time point. The setup in [1] is different: the temporal component is LTL, but $\mathcal{ALC}$ is replaced with the lightweight DL DL-Lite$_{\text{bool}}$. Temporal operators can be applied to concept descriptions, TBoxes, and ABoxes. Here, it is the weak expressive power of the DL component that is responsible for decidability (more precisely, ExpSpace-completeness) of reasoning also in the presence of rigid roles. In the same paper, it is shown that concept subsumption w.r.t. global TBoxes and with rigid roles is undecidable already in the lightweight description logic $\mathcal{EL}$, which provides only for the constructors conjunction and existential restriction.

Our contribution

In this paper, we follow a different approach for regaining decidability in the presence of rigid roles: temporal operators are allowed to occur only in front of axioms (i.e., ABox assertions and TBox axioms), but not as concept constructors. We show that reasoning becomes simpler in this setting: with rigid roles, satisfiability is decidable (more precisely: 2-ExpTime-complete); without rigid roles, the complexity decreases to NExpTime-complete; and without any rigid symbols, it decreases further to ExpTime-complete (i.e., the same complexity as reasoning in $\mathcal{ALC}$ alone). We also consider two other ways of decreasing the complexity of satisfiability to ExpTime. On the one hand, satisfiability without rigid roles (but with rigid concepts) becomes ExpTime-complete if GCIs can occur only as global axioms that must hold in every temporal world. Note that, in this case, ABox assertions are *not* assumed to be global, i.e., the valid ABox assertions may vary over time. On the other hand, satisfiability with rigid concepts and roles becomes ExpTime-complete if the temporal component is restricted appropriately by replacing the temporal operators until (U) and next (X) of LTL with diamond ($\Diamond$), which expresses "sometime in the future."

The situation we concentrate on in this paper (i.e., where temporal operators are allowed to occur only in front of axioms) has been considered before only for the case where there are no rigid concepts or roles. The combination approach introduced in [11] yields a decision procedure for this case, whose worst-case complexity is, however, non-optimal. Our ExpTime upper bound for this case actually also follows from more general results in [12] (see the remark following Theorem 14.15 on page 605 of [12]). However, also in [12], the setting where temporal operators are allowed to occur only in front of axioms is considered only in the absence of rigid symbols.

Obviously, the temporalized DLs we investigate in this paper cannot be used to define temporal concepts such as (1) for concussion with no loss of consciousness. However, they are nevertheless useful in ontology-based applications since they can be used to reason about a temporal sequence of ABoxes w.r.t. a global TBox.[4] For example, in an emergency ward, the vital parameters of a patient are monitored in short intervals (sometimes not longer than 10 minutes), and additional information is available from the patient record and added by doctors and nurses. Using concepts defined in a medical ontology like SNOMED CT, a high-level view of the medical status of the patient at a given time point can be given by an ABox. Obviously, the sequence of ABoxes obtained this way can be described using temporalized ABox assertions. Critical situations, which require the intervention of a doctor, can then be described by a formula in our temporalized DL, and recognized using the reasoning procedures developed in this paper. For example, given a formula ϕ encoding a sequence of ABoxes describing the medical status of Bob, starting at some time point t_0, and the formula ψ defined in (2), we can check whether Bob sometime after t_0 had a concussion with no loss of consciousness by testing $\phi \wedge \neg\psi$ for unsatisfiability.

2 Basic definitions

The temporalized DL $\mathcal{ALC}$-LTL introduced in this paper combines the basic DL $\mathcal{ALC}$ [19] with linear temporal logic (LTL) [16]. We start by recalling the relevant definitions for $\mathcal{ALC}$.

Definition 2.1. *Let N_C, N_R, and N_I respectively be disjoint sets of* concept names, role names, *and* individual names. *The set of* $\mathcal{ALC}$-concept descriptions *is the smallest set such that*

- *all concept names are $\mathcal{ALC}$-concept descriptions;*
- *if C and D are $\mathcal{ALC}$-concept descriptions, then so are $\neg C$, $C \sqcup D$, and $C \sqcap D$;*
- *if C is a $\mathcal{ALC}$-concept description and $r \in N_R$, then $\exists r.C$ and $\forall r.C$ are $\mathcal{ALC}$-concept descriptions.*

A general concept inclusion axiom (GCI) *is of the form $C \sqsubseteq D$, where C, D are $\mathcal{ALC}$-concept descriptions, and an* assertion *is of the form $a : C$ or $(a, b) : r$ where C is an $\mathcal{ALC}$-concept description, r is a role name, and a, b are individual names. We call both GCIs and assertions* $\mathcal{ALC}$-axioms. *A Boolean combination of $\mathcal{ALC}$-axioms is called a* Boolean $\mathcal{ALC}$-knowledge base, *i.e.,*

- *every $\mathcal{ALC}$-axiom is a Boolean $\mathcal{ALC}$-knowledge base;*

[4] Our results also cover the case of temporal TBox axioms, but currently we believe that temporalizing TBox axioms is of less practical relevance than temporalizing ABox assertions.

- *if $\mathcal{B}_1$ and $\mathcal{B}_2$ are Boolean $\mathcal{ALC}$-knowledge bases, then so are $\mathcal{B}_1 \wedge \mathcal{B}_2$, $\mathcal{B}_1 \vee \mathcal{B}_2$, and $\neg\mathcal{B}_1$.*

*An $\mathcal{ALC}$-*TBox *is a conjunction of GCIs, and an $\mathcal{ALC}$-*ABox *is a conjunction of assertions.*

According to this definition, TBoxes and ABoxes are special kinds of Boolean knowledge bases. However, note that they are often written as sets of axioms rather than as conjunctions of these axioms.

The semantics of $\mathcal{ALC}$ is defined through the notion of an interpretation.

Definition 2.2. *An* interpretation *is a pair $\mathcal{I} = (\Delta^\mathcal{I}, \cdot^\mathcal{I})$ where the* domain $\Delta^\mathcal{I}$ *is a non-empty set, and $\cdot^\mathcal{I}$ is a function that assigns to every concept name A a set $A^\mathcal{I} \subseteq \Delta^\mathcal{I}$, to every role name r a binary relation $r^\mathcal{I} \subseteq \Delta^\mathcal{I} \times \Delta^\mathcal{I}$, and to every individual name a an element $a^\mathcal{I} \in \Delta^\mathcal{I}$. This function is extended to $\mathcal{ALC}$-concept descriptions as follows:*

- $(C \sqcap D)^\mathcal{I} = C^\mathcal{I} \cap D^\mathcal{I}, (C \sqcup D)^\mathcal{I} = C^\mathcal{I} \cup D^\mathcal{I}, (\neg C)^\mathcal{I} = \Delta^\mathcal{I} \setminus C^\mathcal{I}$;
- $(\exists r.C)^\mathcal{I} = \{x \in \Delta^\mathcal{I} \mid$ *there is a* $y \in \Delta^\mathcal{I}$ *with* $(x, y) \in r^\mathcal{I}$ *and* $y \in C^\mathcal{I}\}$;
- $(\forall r.C)^\mathcal{I} = \{x \in \Delta^\mathcal{I} \mid$ *for all* $y \in \Delta^\mathcal{I}$, $(x, y) \in r^\mathcal{I}$ *implies* $y \in C^\mathcal{I}\}$.

The interpretation $\mathcal{I}$ is a model *of the $\mathcal{ALC}$-axioms $C \sqsubseteq D$, $a : C$, and $(a, b) : r$ iff it respectively satisfies $C^\mathcal{I} \subseteq D^\mathcal{I}$, $a^\mathcal{I} \in C^\mathcal{I}$, and $(a^\mathcal{I}, b^\mathcal{I}) \in r^\mathcal{I}$. The notion of a model is extended to Boolean $\mathcal{ALC}$-knowledge bases as follows:*

- *$\mathcal{I}$ is a model of $\mathcal{B}_1 \wedge \mathcal{B}_2$ iff it is a model of both $\mathcal{B}_1$ and $\mathcal{B}_2$;*
- *$\mathcal{I}$ is a model of $\mathcal{B}_1 \vee \mathcal{B}_2$ iff it is a model of $\mathcal{B}_1$ or of $\mathcal{B}_2$;*
- *$\mathcal{I}$ is a model of $\neg\mathcal{B}_1$ iff it is not a model of $\mathcal{B}_1$.*

We say that the Boolean $\mathcal{ALC}$-knowledge base $\mathcal{B}$ is consistent *iff it has a model. The concept description C is* satisfiable *w.r.t. the GCI $D_1 \sqsubseteq D_2$ iff there is a model $\mathcal{I}$ of $D_1 \sqsubseteq D_2$ with $C^\mathcal{I} \neq \emptyset$.*

In Description Logics it is often assumed that the interpretations satisfy the *unique name assumption (UNA)*, i.e., different individual names are interpreted by different elements of the domain.

For LTL, we use the variant with a *non-strict until* (U) and a *next* (X) operator. Instead of first introducing the propositional temporal logic LTL, we directly define our new temporalized DL, called $\mathcal{ALC}$-LTL. The difference to LTL is that $\mathcal{ALC}$-axioms replace propositional letters.

Definition 2.3. *$\mathcal{ALC}$-LTL formulae are defined by induction:*

- *if α is an $\mathcal{ALC}$-axiom, then α is an $\mathcal{ALC}$-LTL formula;*
- *if ϕ, ψ are $\mathcal{ALC}$-LTL formulae, then so are $\phi \wedge \psi$, $\phi \vee \psi$, $\neg\phi$, $\phi \mathsf{U} \psi$, and $\mathsf{X}\phi$.*

As usual, we use true as an abbreviation for $A(a) \vee \neg A(a)$, $\Diamond\phi$ as an abbreviation for $\mathsf{true}\mathsf{U}\phi$ (*diamond*, which should be read as "sometime in the future"), and $\Box\phi$ as an abbreviation for $\neg\Diamond\neg\phi$ (*box*, which should be read as "always in the future").

The semantics of $\mathcal{ALC}$-LTL is based on $\mathcal{ALC}$-LTL structures, which are sequences of $\mathcal{ALC}$-interpretations over the same non-empty domain Δ (constant domain assumption). We assume that every individual name stands for a unique element of Δ, i.e., the interpretation of individual names does not change over time (rigid individual names). As usual in DLs, we also make the unique name assumption.

Definition 2.4. *An* $\mathcal{ALC}$-LTL structure *is a sequence $\mathfrak{I} = (\mathcal{I}_i)_{i=0,1,\ldots}$ of $\mathcal{ALC}$-interpretations $\mathcal{I}_i = (\Delta, \cdot^{\mathcal{I}_i})$ obeying the UNA (called* worlds*) such that $a^{\mathcal{I}_i} = a^{\mathcal{I}_j}$ for all individual names a and all $i, j \in \{0, 1, 2, \ldots\}$. Given an $\mathcal{ALC}$-LTL formula ϕ, an $\mathcal{ALC}$-LTL structure $\mathfrak{I} = (\mathcal{I}_i)_{i=0,1,\ldots}$, and a time point $i \in \{0, 1, 2, \ldots\}$, validity of ϕ in $\mathfrak{I}$ at time i (written $\mathfrak{I}, i \models \phi$) is defined inductively:*

$$\begin{array}{lll}
\mathfrak{I}, i \models C \sqsubseteq D & \textit{iff} & C^{\mathcal{I}_i} \subseteq D^{\mathcal{I}_i} \\
\mathfrak{I}, i \models a : C & \textit{iff} & a^{\mathcal{I}_i} \in C^{\mathcal{I}_i} \\
\mathfrak{I}, i \models (a, b) : r & \textit{iff} & (a^{\mathcal{I}_i}, b^{\mathcal{I}_i}) \in r^{\mathcal{I}_i} \\
\mathfrak{I}, i \models \phi \wedge \psi & \textit{iff} & \mathfrak{I}, i \models \phi \textit{ and } \mathfrak{I}, i \models \psi \\
\mathfrak{I}, i \models \phi \vee \psi & \textit{iff} & \mathfrak{I}, i \models \phi \textit{ or } \mathfrak{I}, i \models \psi \\
\mathfrak{I}, i \models \neg\phi & \textit{iff} & \textit{not } \mathfrak{I}, i \models \phi \\
\mathfrak{I}, i \models \mathsf{X}\phi & \textit{iff} & \mathfrak{I}, i+1 \models \phi \\
\mathfrak{I}, i \models \phi \mathsf{U} \psi & \textit{iff} & \textit{there is } k \geq i \textit{ such that } \; \mathfrak{I}, k \models \psi \textit{ and} \\
 & & \hphantom{\textit{there is } k \geq i \textit{ such that } \;} \mathfrak{I}, j \models \phi \textit{ for all } j, i \leq j < k
\end{array}$$

For some concepts and roles, it is not desirable that their interpretation changes over time. Thus, we will sometimes assume that a subset of the set of concept and role names can be designated as being rigid. We will call the elements of this subset *rigid concept names* and *rigid role names.*

Definition 2.5. *We say that the $\mathcal{ALC}$-LTL structure $\mathfrak{I} = (\mathcal{I}_i)_{i=0,1,\ldots}$* respects rigid concept names (role names) *iff $A^{\mathcal{I}_i} = A^{\mathcal{I}_j}$ ($r^{\mathcal{I}_i} = r^{\mathcal{I}_j}$) holds for all $i, j \in \{0, 1, 2, \ldots\}$ and all rigid concept names A (rigid role names r).*

3 The satisfiability problem in $\mathcal{ALC}$-LTL

Depending on whether rigid concept and role names are considered or not, we obtain different variants of the satisfiability problem.

Definition 3.1. *Let ϕ be an $\mathcal{ALC}$-LTL formula and assume that a subset of the set of concept and role names has been designated as being rigid.*

- *We say that ϕ is* satisfiable w.r.t. rigid names *iff there is an $\mathcal{ALC}$-LTL structure $\mathfrak{I}$ respecting rigid concept and role names such that $\mathfrak{I}, 0 \models \phi$.*
- *We say that ϕ is* satisfiable w.r.t. rigid concepts *iff there is an $\mathcal{ALC}$-LTL structure $\mathfrak{I}$ respecting rigid concept names such that $\mathfrak{I}, 0 \models \phi$.*
- *We say that ϕ is* satisfiable without rigid names *(or simply* satisfiable*) iff there is an $\mathcal{ALC}$-LTL structure $\mathfrak{I}$ such that $\mathfrak{I}, 0 \models \phi$.*

In this paper, we show that the complexity of the satisfiability problem for $\mathcal{ALC}$-LTL strongly depends on which of the above cases one considers. Note that it does not really make sense to consider satisfiability w.r.t. rigid role names, but without rigid concept names, as a separate case when investigating the complexity of the satisfiability problem. In fact, rigid concepts can be simulated by rigid roles: just introduce a new rigid role name r_A for each rigid concept name A, and then replace A by $\exists r_A.\top$.

Another dimension that influences the complexity of the satisfiability problem is whether GCIs occur globally or locally in the formula. Intuitively, a GCI occurs globally if it must hold in every world of the $\mathcal{ALC}$-LTL structure.

Definition 3.2. *We say that ϕ is an $\mathcal{ALC}$-LTL formula* with global GCIs *iff it is of the form $\phi = \Box\mathcal{B} \wedge \varphi$ where $\mathcal{B}$ is a conjunction of $\mathcal{ALC}$-axioms and φ is an $\mathcal{ALC}$-LTL formula that does not contain GCIs. We denote the fragment of $\mathcal{ALC}$-LTL that contains only $\mathcal{ALC}$-LTL formulae with global GCIs by $\mathcal{ALC}$-LTL$|_{gGCI}$.*

Note that saying, in the above definition, that $\mathcal{B}$ is a *conjunction* of $\mathcal{ALC}$-axioms just means that $\mathcal{B}$ is a TBox together with an ABox. We could have restricted $\mathcal{B}$ to being a conjunction of GCIs (i.e., a TBox) since assertions α in $\mathcal{B}$ could be moved as conjuncts $\Box\alpha$ to φ.[5] However, it turns out to be more convenient to allow also ABox assertions to occur in the "global part" $\Box\mathcal{B}$ of ϕ.

Instead of restricting to $\mathcal{ALC}$-LTL formulae with global GCIs, we can also restrict the temporal component, by considering the fragment $\mathcal{ALC}$-LTL$|_\Diamond$ of $\mathcal{ALC}$-LTL in which $\Diamond$ is the only temporal operator. In this fragment, neither U nor X is definable.

Definition 3.3. *$\mathcal{ALC}$-LTL$|_\Diamond$ formulae are defined by induction:*

- *if α is an $\mathcal{ALC}$-axiom, then α is an $\mathcal{ALC}$-LTL$|_\Diamond$ formula;*
- *if ϕ, ψ are $\mathcal{ALC}$-LTL$|_\Diamond$ formulae, then so are $\phi \wedge \psi$, $\phi \vee \psi$, $\neg\phi$, and $\Diamond\phi$.*

[5] This is the reason why we talk about $\mathcal{ALC}$-LTL formulae *with global GCIs* in this case, rather than about $\mathcal{ALC}$-LTL formulae with global axioms.

	W.r.t. rigid names	W.r.t. rigid concepts	Without rigid names
$\mathcal{ALC}$-LTL	2-ExpTime-complete	NExpTime-complete	ExpTime-complete
$\mathcal{ALC}$-LTL$\vert_{gGCI}$	2-ExpTime-complete	ExpTime-complete	ExpTime-complete
$\mathcal{ALC}$-LTL$\vert_{\Diamond}$	ExpTime-complete	ExpTime-complete	ExpTime-complete

Table 1: Complexity of the satisfiability problem in $\mathcal{ALC}$-LTL and its fragments.

The semantics of $\mathcal{ALC}$-LTL$|_{\Diamond}$ formulae is defined as in the case of $\mathcal{ALC}$-LTL. In particular, the interpretation of the diamond operator is defined as

$$\mathfrak{I}, i \models \Diamond\phi \quad \text{iff} \quad \text{there is } k \geq i \text{ such that } \mathfrak{I}, k \models \phi.$$

Table 1 summarizes the results of our investigation of the complexity of the satisfiability problem in $\mathcal{ALC}$-LTL and its fragments. This table shows that the complexity of the satisfiability problem in $\mathcal{ALC}$-LTL increases (from ExpTime, which is the complexity of the satisfiability problem in $\mathcal{ALC}$, to NExpTime) if rigid concepts names are available. The additional presence of rigid role names further increases the complexity to 2-ExpTime. The restriction to $\mathcal{ALC}$-LTL$|_{gGCI}$ (i.e., global GCIs) has no effect on the complexity in the presence of rigid role names. However, it decreases the complexity to ExpTime if only rigid concept names are available. In $\mathcal{ALC}$-LTL$|_{\Diamond}$, the satisfiability problem is only ExpTime-complete even w.r.t. rigid names.

In Section 4, we will show the results for $\mathcal{ALC}$-LTL and $\mathcal{ALC}$-LTL$|_{gGCI}$ for the case of rigid names. Section 5 considers satisfiability in $\mathcal{ALC}$-LTL and in its fragment $\mathcal{ALC}$-LTL$|_{gGCI}$ without rigid names, and Section 6 is concerned with satisfiability in $\mathcal{ALC}$-LTL and $\mathcal{ALC}$-LTL$|_{gGCI}$ w.r.t. rigid concepts. Finally, in Section 7, we consider the satisfiability problem in $\mathcal{ALC}$-LTL$|_{\Diamond}$.

4 Reasoning with rigid names

In this section, we investigate the complexity of the satisfiability problem in $\mathcal{ALC}$-LTL and its fragment $\mathcal{ALC}$-LTL$|_{gGCI}$ if rigid concepts and roles are available.

Theorem 4.1. *Satisfiability in $\mathcal{ALC}$-LTL w.r.t. rigid names is* 2-ExpTime-*complete.*

First, we show 2-ExpTime-hardness.

Lemma 4.2. *Satisfiability in $\mathcal{ALC}$-LTL w.r.t. rigid names is* 2-ExpTime-*hard.*

Proof. The proof is by reduction of the word problem for exponentially space bounded alternating Turing machines (ATMs). An ATM is of the form $\mathcal{M} =$

$(Q, \Sigma, \Gamma, q_0, \Theta)$, where $Q = Q_\exists \uplus Q_\forall \uplus \{q_a, q_r\}$ is a finite set of *states*, partitioned into *existential states* from $Q_\exists$, *universal states* from $Q_\forall$, an *accepting state* q_a, and a *rejecting state* q_r; Σ is the *input alphabet* and $\Gamma \supseteq \Sigma$ the *work alphabet* containing a *blank symbol* $B \notin \Sigma$; $q_0 \in Q_\exists \cup Q_\forall$ is the *initial* state; and the *transition relation* Θ is of the form $\Theta \subseteq Q \times \Gamma \times Q \times \Gamma \times \{L, R\}$. We write $\Theta(q, a)$ for $\{(q', b, M) \mid (q, a, q', b, M) \in \Theta\}$.

A *configuration* of an ATM is a word wqw' with $w, w' \in \Gamma^*$ and $q \in Q$. The intended meaning is that the (one-sided infinite) tape contains the word ww' with only blanks behind it, the machine is in state q, and the head is on the left-most symbol of w'. The *successor configurations* of a configuration wqw' are defined in the usual way in terms of the transition relation Θ. A *halting configuration* is of the form wqw' with $q \in \{q_a, q_r\}$. We may assume w.l.o.g. that any configuration other than a halting configuration has at least one successor configuration. A *computation* of an ATM $\mathcal{M}$ on a word w is a (finite or infinite) sequence of successive configurations $K_1, K_2, \ldots$. For the ATMs considered here, we may assume without loss of generality that they have only finite computations on any input. Since this case is simpler than the general one, we define acceptance for ATMs with finite computations and refer to [10] for the full definition. Let $\mathcal{M}$ be such an ATM. A halting configuration is *accepting* iff it is of the form wq_aw'. For other configurations $K = wqw'$, the acceptance behaviour depends on q: if $q \in Q_\exists$, then K is accepting iff at least one successor configuration is accepting; if $q \in Q_\forall$, then K is accepting iff all successor configurations are accepting. Finally, the ATM $\mathcal{M}$ with initial state q_0 *accepts* the input w iff the *initial configuration* q_0w is accepting. We use $L(\mathcal{M})$ to denote the language accepted by $\mathcal{M}$, i.e.,

$$L(\mathcal{M}) = \{w \in \Sigma^* \mid \mathcal{M} \text{ accepts } w\}.$$

The *word problem* for $\mathcal{M}$ is the following decision problem: given a word $w \in \Sigma^*$, does $w \in L(\mathcal{M})$ hold or not?

There exists an exponentially space bounded ATM $\mathcal{M} = (Q, \Sigma, \Gamma, q_0, \Theta)$ whose word problem is 2-ExpTime-hard [10]. Our aim is to reduce the word problem for this ATM $\mathcal{M}$ to satisfiability in $\mathcal{ALC}$-LTL w.r.t. rigid names. We may assume that the length of every computation of $\mathcal{M}$ on $w \in \Sigma^k$ is bounded by 2^{2^k}, and all the configurations wqw' in such computations satisfy $|ww'| \leq 2^k$. We may also assume w.l.o.g. that $\mathcal{M}$ never attempts to move to the left when it is on the left-most tape cell.

Let $w = \sigma_0 \cdots \sigma_{k-1} \in \Sigma^*$ be an input to $\mathcal{M}$. We construct an $\mathcal{ALC}$-LTL formula $\phi_{\mathcal{M},w}$ such that $w \in L(\mathcal{M})$ iff $\phi_{\mathcal{M},w}$ is satisfiable w.r.t. rigid names. In an $\mathcal{ALC}$-LTL structure satisfying $\phi_{\mathcal{M},w}$, each domain element from Δ describes a single tape cell of a configuration of $\mathcal{M}$. We use the following symbols:

- a single individual name a that identifies the first tape cell of the first configuration;

- a single *rigid* role name r to represent "going to the next tape cell in the same configuration" and "going from the last tape cell in a configuration to the first cell in a successor configuration";

- the elements of Q and Γ are viewed as *rigid* concept names;

- *rigid* concept names $A_0, \ldots, A_{k-1}$ are the bits of a binary counter that numbers the tape cells in each configuration;

- auxiliary *rigid* concept name I and H; I indicates the initial configuration and H indicates that, in the current configuration, the head is to the left of the current tape cell;

- auxiliary *rigid* concept names $T_{q,\sigma,M}$ for all $q \in Q$, $\sigma \in \Gamma$, and $M \in \{L, R\}$; intuitively, $T_{q,\sigma,M}$ is true if, in the current configuration, the head is on the left neighboring cell and the machine executes transition (q, σ, M);

- for each element of Q and Γ, a non-rigid concept name which is distinguished from its rigid version by a prime;

- (non-rigid) concept names $A'_0, \ldots, A'_{k-1}$ to realize an orthogonal counter (in the sense that it counts along the temporal dimension instead of along r).

Before giving the formal reduction, let us explain the underlying intuition. As said above, a single configuration is described as a sequence of r-successors of length 2^k of the individual representing its first tape cell. The tape cells of a configuration are numbered from 0 to $2^k - 1$, using the counter realized through the concept names $A_0, \ldots, A_{k-1}$. We denote the concept that expresses that the counter has value $i, 0 \leq i < 2^k$, by $(C_A = i)$; i.e., $(C_A = 0)$ denotes $\neg A_0 \sqcap \neg A_1 \sqcap \ldots \sqcap \neg A_{k-1}$, $(C_A = 1)$ denotes $A_0 \sqcap \neg A_1 \sqcap \ldots \sqcap \neg A_{k-1}$, ..., $(C_A = 2^k - 1)$ denotes $A_0 \sqcap A_1 \sqcap \ldots \sqcap A_{k-1}$.

The r-successor of the last tape cell of a given configuration represents the first tape cell of a successor configuration of this configuration. It obtains the number 0, i.e., the counter realized by $A_0, \ldots, A_{k-1}$ is reset to 0, which simply means that we count modulo 2^k. Since we have an *alternating* Turing machine, it is not enough to consider one sequence of configurations. For a configuration with a universal state, we must consider all successor configurations. Thus, we do not consider a single sequence of r-successors, but rather a tree of r-successors.

The main problem to solve when defining the reduction is to ensure that each configuration following a given configuration in the tree of r-successors is actually a successor configuration, i.e., tape cells that are not immediately to the left or right of the head remain unchanged, and the other tape cells are changed according to the transition relation. For the first type of cells this means that, given a cell numbered i in the current configuration, the next cell with the same number should carry the same symbol. However, we cannot remember the value i of the A-counter when going down along the sequence of r-successors since this counter is incremented (modulo 2^k) when going to an r-successor. This is where the temporal dimension comes into play. Here, we realize an A'-counter, using the (non-rigid) concept names $A'_0, \ldots, A'_{k-1}$, whose value does not change along the r-dimension, but is incremented (modulo 2^k) along the temporal dimension. This additional counter, together with the non-rigid copies of the symbols from Q and Γ, can be used to transfer a symbol from a tape cell in a given configuration to the corresponding tape cell in a successor configuration (see below).

In the following, we use $\phi \rightarrow \psi$ as an abbreviation for $\neg\phi \vee \psi$, $C \Rightarrow D$ as an abbreviation for $\neg C \sqcup D$, and $C \Leftrightarrow D$ as an abbreviation for $(C \Rightarrow D) \sqcap (D \Rightarrow C)$.

The reduction formula $\phi_{\mathcal{M},w}$ is the conjunction of the following formulae:
We start by setting up I, H, r, and the A-counter:

- I behaves as described, i.e., it marks the initial configuration, whose first tape cell is represented by the individual a:

$$\Box\,(a : I)$$
$$\Box\,\big(I \sqcap \neg(C_A = 2^k - 1) \sqsubseteq \forall r.I\big)$$

- H behaves as described, i.e., it marks the tape cells that are to the right of the head, where the head position is indicated by having a state concept at this cell:

$$\Box\left((H \sqcup \bigsqcup_{q\in Q} q) \sqcap \neg(C_A = 2^k - 1) \sqsubseteq \forall r.H\right)$$

- there is always an r-successor, except when we meet the head in a halting configuration:

$$\Box\,(\neg(q_a \sqcup q_r) \sqsubseteq \exists r.\top)$$

- the counter A-counter realized by $A_0, \ldots, A_{k-1}$ has value 0 at a, and it is incremented along r (modulo 2^k):

$$\Box\,(a : (C_A = 0))$$
$$\Box\left(\top \sqsubseteq \bigsqcap_{i<k}\big(\bigsqcap_{j<i} A_j\big) \Rightarrow \big((A_i \Rightarrow \forall r.\neg A_i) \sqcap (\neg A_i \Rightarrow \forall r.A_i)\big)\right)$$
$$\Box\left(\top \sqsubseteq \bigsqcap_{i<k}\big(\bigsqcup_{j<i} \neg A_j\big) \Rightarrow \big((A_i \Rightarrow \forall r.A_i) \sqcap (\neg A_i \Rightarrow \forall r.\neg A_i)\big)\right)$$

Some properties of runs of ATMs can be formalized without using the temporal dimension:

- The initial configuration is the one induced by the input $w = \sigma_0 \dots \sigma_{k-1}$:

$$\begin{aligned}
\Box\,(a : \forall r^i.\sigma_i) &\qquad\qquad \text{for } i < k \\
\Box\,\big(a : \forall r^k.B\big) & \\
\Box\,\big(I \sqcap B \sqcap \neg(C_A = 2^k - 1) \sqsubseteq \forall r.B\big) &
\end{aligned}$$

- The computation starts on the left-most tape cell of this initial configuration in state q_0:

$$\Box\,(a : q_0)$$

- Each tape cell is labelled with exactly one symbol and at most one state:

$$\Box\left(\top \sqsubseteq \bigsqcup_{\sigma\in\Gamma}(\sigma \sqcap \neg \bigsqcap_{\sigma'\in\Gamma\setminus\{\sigma\}} \neg\sigma')\right)$$

$$\Box\left(\top \sqsubseteq \bigsqcap_{q,q'\in Q, q\neq q'} \neg(q \sqcap q')\right)$$

- There is only one head position per configuration:

$$\Box\left(H \sqsubseteq \bigsqcap_{q\in Q} \neg q\right)$$

It remains to implement the transitions and to say that symbols not under the head do not change in successor configurations. Here we need the temporal dimension. We start with setting up the A'-counter:

- for every value of the A'-counter realized using the (non-rigid) concept names $A'_0, \dots, A'_{k-1}$, there is a time point at which a has that value:

$$\Box\left(\textstyle\bigwedge_{i<k}\big(\bigwedge_{j<i} a : A'_j\big) \rightarrow \big((a : A'_i \rightarrow \mathsf{X} a : \neg A'_i) \wedge (a : \neg A'_i \rightarrow \mathsf{X} a : A'_i)\big)\right)$$

$$\Box\left(\textstyle\bigwedge_{i<k}\big(\bigvee_{j<i} a : \neg A'_j\big) \rightarrow \big((a : A'_i \rightarrow \mathsf{X} a : A'_i) \wedge (a : \neg A'_i \rightarrow \mathsf{X} a : \neg A'_i)\big)\right)$$

  This is basically the same formula as for the A-counter, but the values of the A'-counter are considered for the fixed initial individual a, and they are incremented along the temporal dimension.

- The value of the A'-counter is preserved along r, i.e., for all $i, 0 \leq i < k$, we require:

$$\Box\,(A'_i \sqsubseteq \forall r.A'_i)$$

$$\Box\,(\neg A'_i \sqsubseteq \forall r.\neg A'_i)$$

In summary, we have associated one "temporal slice" with each counter value of the second counter. In the following, we use $(C_A = C_{A'})$ to denote the concept $(A_0 \Leftrightarrow A'_0) \sqcap \ldots \sqcap (A_{k-1} \Leftrightarrow A'_{k-1})$, which states that the value of the A-counter coincides with the value of the A'-counter. Accordingly, $(C_A = C_{A'} + 1 \bmod 2^k)$ expresses that the value of the A-counter is equal to the value of the A'-counter plus 1 (modulo 2^k). This can be expressed by a recasting of the incrementation concept given already twice above:

$$
\begin{aligned}
(C_A = C_{A'} + 1 \bmod 2^k) := \; & \bigsqcap_{i<k} \big(\bigsqcap_{j<i} A'_j \big) \Rightarrow \big((A'_i \Rightarrow \neg A_i) \sqcap (\neg A'_i \Rightarrow A_i) \big) \sqcap \\
& \bigsqcap_{i<k} \big(\bigsqcup_{j<i} \neg A'_j \big) \Rightarrow \big((A'_i \Rightarrow A_i) \sqcap (\neg A'_i \Rightarrow \neg A_i) \big)
\end{aligned}
$$

The concept $(C_A = C_{A'} + 2 \bmod 2^k)$, which expresses that the value of the A-counter is equal to the value of the A'-counter plus 2 (modulo 2^k), can be defined similarly, using an auxiliary set $A''_0, \ldots, A''_{k-1}$ of non-rigid concept names.

- We can now say that symbols not under the head do not change:

$$
\begin{array}{ll}
\Box \left(\sigma \sqcap \bigsqcap_{q \in Q} \neg q \sqcap (C_A = C_{A'}) \sqsubseteq \forall r.\sigma' \right) & \text{for all } \sigma \in \Gamma \\
\Box \, (\sigma' \sqcap \neg (C_A = C_{A'}) \sqsubseteq \forall r.\sigma') & \text{for all } \sigma \in \Gamma \\
\Box \, (\sigma' \sqcap (C_A = C_{A'}) \sqsubseteq \sigma) & \text{for all } \sigma \in \Gamma
\end{array}
$$

- Transitions are implemented in a similar way. The fact that we have an *alternating* Turing is taken into account by enforcing a branching on universal transitions:

$$
\begin{array}{ll}
\Box \left(q \sqcap \sigma \sqsubseteq \bigsqcup_{(p,\nu,M) \in \Theta(q,\sigma)} \forall r.T_{p,\nu,M} \right) & \text{for all } q \in Q_\exists, \sigma \in \Sigma \\
\Box \left(q \sqcap \sigma \sqsubseteq \bigsqcap_{(p,\nu,M) \in \Theta(q,\sigma)} \exists r.T_{p,\nu,M} \right) & \text{for all } q \in Q_\forall, \sigma \in \Sigma \\
\Box \big(T_{q,\sigma,M} \sqcap (C_A = C_{A'} + 1 \bmod 2^k) \sqsubseteq \forall r.\sigma' \big) & \text{for all } \sigma \in \Gamma, q \in Q, \\
 & M \in \{L, R\} \\
\Box \, (T_{q,\sigma,R} \sqcap (C_A = C_{A'}) \sqsubseteq \forall r.q') & \text{for all } \sigma \in \Gamma, q \in Q \\
\Box \big(T_{q,\sigma,L} \sqcap (C_A = C_{A'} + 2 \bmod 2^k) \sqsubseteq \forall r.q' \big) & \text{for all } \sigma \in \Gamma, q \in Q \\
\Box \, (q' \sqcap \neg (C_A = C_{A'}) \sqsubseteq \forall r.q') & \text{for all } q \in Q \\
\Box \, (q' \sqcap (C_A = C_{A'}) \sqsubseteq q) & \text{for all } q \in Q
\end{array}
$$

It remains to encode the fact that the input $w = \sigma_0 \ldots \sigma_{k-1}$ is accepted. Since any computation of $\mathcal{M}$ is terminating, and halting configurations (i.e., configurations with state q_a or q_r) are the only ones without successor configurations, this can be done as follows:

- We can express the fact that the initial configuration for input w is accepting by disallowing the state q_r to occur:

$$\Box\,(\top \sqsubseteq \neg q_r)$$

This finishes the definition of the $\mathcal{ALC}$-LTL formula $\phi_{\mathcal{M},w}$, which is the conjunction of the formulae introduced above. It is easy to see that the size of $\phi_{\mathcal{M},w}$ is polynomial in k, and that $\phi_{\mathcal{M},w}$ is satisfiable w.r.t. rigid names iff $w \in L(\mathcal{M})$. ❏

Next, we show that the complexity lower bound provided by the above lemma is tight.

Lemma 4.3. *Satisfiability in $\mathcal{ALC}$-LTL w.r.t. rigid names is in* 2-ExpTime.

Proof. Let ϕ be an $\mathcal{ALC}$-LTL formula. We build its propositional abstraction $\widehat{\phi}$ by replacing each $\mathcal{ALC}$-axiom by a propositional variable such that there is a 1–1 relationship between the $\mathcal{ALC}$-axioms $\alpha_1, \ldots, \alpha_n$ occurring in ϕ and the propositional variables $p_1, \ldots, p_n$ used for the abstraction. We assume in the following that p_i was used to replace α_i ($i = 1, \ldots, n$).

Consider a set $\mathcal{S} \subseteq \mathcal{P}(\{p_1, \ldots, p_n\})$, i.e., a set of subsets of $\{p_1, \ldots, p_n\}$. Such a set induces the following (propositional) LTL formula:

$$\widehat{\phi}_{\mathcal{S}} := \widehat{\phi} \wedge \Box \left(\bigvee_{X \in \mathcal{S}} \left(\bigwedge_{p \in X} p \wedge \bigwedge_{p \notin X} \neg p \right) \right)$$

If ϕ is satisfiable in an $\mathcal{ALC}$-LTL structure $\mathfrak{I} = (\mathcal{I}_i)_{i=0,1,\ldots}$, then there is an $\mathcal{S} \subseteq \mathcal{P}(\{p_1, \ldots, p_n\})$ such that $\widehat{\phi}_{\mathcal{S}}$ is satisfiable in a propositional LTL structure. In fact, for each $\mathcal{ALC}$-interpretation $\mathcal{I}_i$ of $\mathfrak{I}$, we define the set

$$X_i := \{p_j \mid 1 \leq j \leq n \text{ and } \mathcal{I}_i \text{ satisfies } \alpha_j\},$$

and then take $\mathcal{S} = \{X_i \mid i = 0, 1, \ldots\}$. The fact that $\mathfrak{I}$ satisfies ϕ implies that its propositional abstraction satisfies $\widehat{\phi}_{\mathcal{S}}$, where the propositional abstraction $\widehat{\mathfrak{I}} = (w_i)_{i=0,1,\ldots}$ of $\mathfrak{I}$ is defined such that world w_i makes variable p_j true iff $\mathcal{I}_i$ satisfies α_j. However, guessing such a set $\mathcal{S} \subseteq \mathcal{P}(\{p_1, \ldots, p_n\})$ and then testing whether the induced propositional LTL formula $\widehat{\phi}_{\mathcal{S}}$ is satisfiable is not sufficient for checking satisfiability w.r.t. rigid names of the $\mathcal{ALC}$-LTL formula ϕ. We must also check whether the guessed set $\mathcal{S}$ can indeed be induced by some $\mathcal{ALC}$-LTL structure that respects the rigid concept and role names.

To this purpose, assume that a set $\mathcal{S} = \{X_1, \ldots, X_k\} \subseteq \mathcal{P}(\{p_1, \ldots, p_n\})$ is given. For every $i, 1 \leq i \leq k$, and every *flexible* concept name A (*flexible* role name r) occurring in $\alpha_1, \ldots, \alpha_n$, we introduce a copy $A^{(i)}$ ($r^{(i)}$). We call $A^{(i)}$ ($r^{(i)}$) the ith copy of A (r). The $\mathcal{ALC}$-axiom $\alpha_j^{(i)}$ is obtained from α_j by replacing every occurrence of a flexible name by its ith copy. The sets X_i ($1 \leq i \leq k$) induce the following Boolean $\mathcal{ALC}$-knowledge bases:

$$\mathcal{B}_i := \bigwedge_{p_j \in X_i} \alpha_j^{(i)} \wedge \bigwedge_{p_j \notin X_i} \neg \alpha_j^{(i)}$$

Claim. *The $\mathcal{ALC}$-LTL formula ϕ is satisfiable w.r.t. rigid names iff there is a set $\mathcal{S} = \{X_1, \ldots, X_k\} \subseteq \mathcal{P}(\{p_1, \ldots, p_n\})$ such that the propositional LTL formula $\widehat{\phi}_{\mathcal{S}}$ is satisfiable and the Boolean $\mathcal{ALC}$-knowledge base $\mathcal{B} := \bigwedge_{1 \leq i \leq k} \mathcal{B}_i$ is consistent.*

For the "only if" direction, recall that we have already seen how an $\mathcal{ALC}$-LTL structure $\mathfrak{I} = (\mathcal{I}_\iota)_{\iota=0,1,\ldots}$ satisfying ϕ can be used to define a set $\mathcal{S} \subseteq \mathcal{P}(\{p_1, \ldots, p_n\})$ such that $\widehat{\phi}_{\mathcal{S}}$ is satisfiable. Let $\mathcal{S} = \{X_1, \ldots, X_k\}$. For each $\iota = 0, 1, \ldots$ there is an index $i_\iota \in \{1, \ldots, k\}$ such that $\mathcal{I}_\iota$ induces the set X_{i_ι}, i.e.,

$$X_{i_\iota} = \{p_j \mid 1 \leq j \leq n \text{ and } \mathcal{I}_\iota \text{ satisfies } \alpha_j\},$$

and, conversely, for each $i \in \{1, \ldots, k\}$ there is an index $\iota \in \{0, 1, 2, \ldots\}$ such that $i = i_\iota$. Let $\iota_1, \ldots, \iota_k \in \{0, 1, 2, \ldots\}$ be such that $i_{\iota_1} = 1, \ldots, i_{\iota_k} = k$. The $\mathcal{ALC}$-interpretation $\mathcal{J}_i$ is obtained from $\mathcal{I}_{\iota_i}$ by interpreting the ith copy of each flexible name like the original flexible name, and by forgetting about the interpretations of the flexible names. By our construction of $\mathcal{J}_i$ and our definition of the Boolean $\mathcal{ALC}$-knowledge base $\mathcal{B}_i$, we have that $\mathcal{J}_i$ is a model of $\mathcal{B}_i$. Recall that the interpretations $\mathcal{I}_{\iota_1}, \ldots, \mathcal{I}_{\iota_k}$ (and thus also $\mathcal{J}_1, \ldots, \mathcal{J}_k$) all have the same domain. In addition, the interpretations of the rigid names coincide in $\mathcal{I}_{\iota_1}, \ldots, \mathcal{I}_{\iota_k}$ (and thus also in $\mathcal{J}_1, \ldots, \mathcal{J}_k$) and the flexible symbols have been renamed. Thus, the union $\mathcal{J}$ of $\mathcal{J}_1, \ldots, \mathcal{J}_k$ is a well-defined $\mathcal{ALC}$-interpretation, and it is easy to see that it is a model of $\mathcal{B} = \bigwedge_{1 \leq i \leq k} \mathcal{B}_i$.

To show the "if" direction, assume that there is a set $\mathcal{S} = \{X_1, \ldots, X_k\} \subseteq \mathcal{P}(\{p_1, \ldots, p_n\})$ such that $\widehat{\phi}_{\mathcal{S}}$ is satisfiable and $\mathcal{B} := \bigwedge_{1 \leq i \leq k} \mathcal{B}_i$ is consistent. Let $\widehat{\mathfrak{I}} = (w_\iota)_{\iota=0,1,\ldots}$ be a propositional LTL structure satisfying $\widehat{\phi}_{\mathcal{S}}$, and let $\mathcal{J}$ be an $\mathcal{ALC}$-interpretation satisfying $\mathcal{B}$. By the definition of $\widehat{\phi}_{\mathcal{S}}$, for every world w_ι there is exactly one index $i_\iota \in \{1, \ldots, k\}$ such that w_ι satisfies

$$\bigwedge_{p \in X_{i_\iota}} p \wedge \bigwedge_{p \notin X_{i_\iota}} \neg p.$$

For $i \in \{1, \ldots, k\}$, we use the $\mathcal{ALC}$-interpretation $\mathcal{J}$ satisfying $\mathcal{B}$ to define an $\mathcal{ALC}$-interpretation $\mathcal{J}_i$ as follows: $\mathcal{J}_i$ interprets the rigid names like $\mathcal{J}$, and it interprets the flexible names just as $\mathcal{J}$ interprets the ith copies of them. Note that the interpretations $\mathcal{J}_i$ are over the same domain and respect the rigid symbols, i.e., they interpret them identically. We can now define an $\mathcal{ALC}$-LTL structure respecting rigid symbols and satisfying ϕ as follows: $\mathfrak{I} := (\mathcal{I}_\iota)_{\iota=0,1,\ldots}$ where $\mathcal{I}_\iota := \mathcal{J}_{i_\iota}$.

This completes the proof of the claim. It remains to show that the claim provides us with a decision procedure for satisfiability in $\mathcal{ALC}$-LTL w.r.t. rigid names that runs in deterministic double-exponential time.

First, note that there are 2^{2^n} many subsets $\mathcal{S}$ of $\mathcal{P}(\{p_1,\ldots,p_n\})$ to be tested, where n is of course linearly bounded by the size of ϕ. For each of these subsets $\mathcal{S} = \{X_1,\ldots,X_k\}$, whose cardinality k is bounded by 2^n, we need to check satisfiability of $\widehat{\phi}_\mathcal{S}$ and consistency of $\mathcal{B} = \bigwedge_{1\le i\le k}\mathcal{B}_i$.

The size of $\widehat{\phi}_\mathcal{S}$ is at most exponential in the size of ϕ, and the complexity of the satisfiability problem in propositional LTL is in PSPACE, and thus in particular in EXPTIME. Consequently, satisfiability of $\widehat{\phi}_\mathcal{S}$ can be tested in double-exponential time in the size of ϕ.

The Boolean $\mathcal{ALC}$-knowledge base $\mathcal{B}$ is a conjunction of $k \le 2^n$ Boolean $\mathcal{ALC}$-knowledge bases $\mathcal{B}_i$, where the size of each $\mathcal{B}_i$ is polynomial in the size of ϕ. The consistency problem for Boolean $\mathcal{ALC}$-knowledge base is EXPTIME-complete (see, e.g., Theorem 2.27 in [12]). Consequently, consistency of $\mathcal{B}$ can also be tested in double-exponential time in the size of the input formula ϕ.

Overall, we thus have double-exponentially many tests, where each test takes double-exponential time. This provides us with a double-exponential bound for testing satisfiability in $\mathcal{ALC}$-LTL w.r.t. rigid names based on the above claim. ❏

This 2-EXPTIME upper bound obviously also applies to the restricted case where only global GCIs are available. Looking back at the proof of 2-EXPTIME-hardness (Lemma 4.2), it is easy to see that all the GCIs used there are actually global. To be more precise, the formula $\phi_{\mathcal{M},w}$ constructed in the proof of Lemma 4.2 is of the form $(\bigwedge_i \Box\alpha_i) \wedge \psi$, where (i) the α_i are GCIs or assertions, and (ii) ψ is an $\mathcal{ALC}$-LTL formula that does not contain GCIs. Since $(\bigwedge_i \Box\alpha_i)\wedge\psi$ is equivalent to $\Box\,(\bigwedge_i \alpha_i)\wedge\psi$, this shows that satisfiability in $\mathcal{ALC}$-LTL$|_{gGCI}$ w.r.t. rigid names is also 2-EXPTIME-hard.

Corollary 4.4. *Satisfiability in $\mathcal{ALC}$-LTL$|_{gGCI}$ w.r.t. rigid names is* 2-EXPTIME-*complete.*

5 Reasoning without rigid names

In this section, we consider the case where we have no rigid names at all. As mentioned in the introduction, this case is also treated in [12], where it is shown that an ExpTime upper bound for the satisfiability problem follows from more general results proved in Chapter 11 of [12] (see the remark following Theorem 14.15 on page 605 of [12]). For the sake of completeness, we give a direct proof of this upper bound below. To this purpose, we will show that, in this simple case, the claim shown in the proof of Lemma 4.3 implies that satisfiability can be decided in deterministic exponential time.

If we are interested in satisfiability without rigid names, then all role and concept names are assumed to be flexible. Consequently, the Boolean $\mathcal{ALC}$-knowledge bases $\mathcal{B}_i$ defined in the proof of Lemma 4.3 do not share concept or role names, and can thus be tested for consistency separately.

Lemma 5.1. *Let $\mathcal{B}_1, \ldots, \mathcal{B}_k$ be Boolean $\mathcal{ALC}$-knowledge bases over disjoint sets of names. Then $\mathcal{B}_1 \wedge \ldots \wedge \mathcal{B}_k$ is consistent iff, for each $i = 1, \ldots, k$, $\mathcal{B}_i$ is consistent.*

Proof. Obviously, consistency of $\mathcal{B}_1 \wedge \ldots \wedge \mathcal{B}_k$ implies consistency of $\mathcal{B}_i$ for all $i, 1 \leq i \leq k$. Conversely, if all the knowledge bases $\mathcal{B}_i$ $(i = 1, \ldots, k)$ are consistent, then each of them has a model with a countably infinite domain. This means that we can assume without loss of generality that these models have the same domain. In addition, since these models obey the UNA, we can also assume that they interpret the individual names in the same way. Putting together the interpretations of all concept and role names from the separate models yields an interpretation that is a model of all the knowledge bases $\mathcal{B}_1, \ldots, \mathcal{B}_k$, and thus a model of $\mathcal{B}_1 \wedge \ldots \wedge \mathcal{B}_k$. ❏

Looking back at the proof of Lemma 4.3, we see that k is exponential in the size of the input formula ϕ, and that each Boolean $\mathcal{ALC}$-knowledge bases $\mathcal{B}_i$ has a size that is polynomial in the size of ϕ. Thus, the consistency test for each $\mathcal{B}_i$ takes time exponential in the size of ϕ. Consequently, testing all the knowledge bases $\mathcal{B}_1, \ldots, \mathcal{B}_k$ for consistency can be achieved in exponential time.

However, if we simply apply the decision procedure suggested by the claim from the proof of Lemma 4.3, we do not obtain an ExpTime-procedure. In fact, guessing a subset $\mathcal{S} \subseteq \mathcal{P}(\{p_1, \ldots, p_n\})$ would require non-deterministic exponential time (since we have exponentially many sets to choose from), and testing the propositional LTL formula $\widehat{\phi}_{\mathcal{S}}$ for satisfiability would require exponential space (since the size of $\widehat{\phi}_{\mathcal{S}}$ can be exponential in the size of ϕ, and satisfiability in propositional LTL is PSpace-complete).

The first problem can easily be avoided. Instead of guessing an appropriate set $\mathcal{S}$, we compute the maximal one: let $\widehat{\mathcal{S}}$ consist of all sets $X \subseteq \{p_1, \ldots, p_n\}$ such that the Boolean $\mathcal{ALC}$-knowledge base

$$\mathcal{B}_X := \bigwedge_{p_j \in X} \alpha_j \wedge \bigwedge_{p_j \notin X} \neg \alpha_j$$

is consistent. Note that we need not rename flexible names here since the knowledge bases $\mathcal{B}_X$ are considered separately.

Lemma 5.2. *The $\mathcal{ALC}$-LTL formula ϕ is satisfiable iff the propositional LTL formula $\widehat{\phi}_{\widehat{\mathcal{S}}}$ is satisfiable.*

Proof. The "if" direction is an immediate consequence of Lemma 5.1, the claim shown in the proof of Lemma 4.3, and the definition of $\widehat{\mathcal{S}}$.

For the "only if" direction, assume that ϕ is satisfiable. By the claim shown in the proof of Lemma 4.3, this implies that there is a set $\mathcal{S} = \{X_1, \ldots, X_k\} \subseteq \mathcal{P}(\{p_1, \ldots, p_n\})$ such that $\widehat{\phi}_{\mathcal{S}}$ is satisfiable and $\bigwedge_{1 \leq i \leq k} \mathcal{B}_i$ is consistent. Consistency of $\bigwedge_{1 \leq i \leq k} \mathcal{B}_i$ implies that the knowledge bases $\mathcal{B}_{X_i}$ $(i = 1, \ldots, k)$ are consistent, and thus we have $\mathcal{S} \subseteq \widehat{\mathcal{S}}$. Consequently, satisfiability of $\widehat{\phi}_{\mathcal{S}}$ implies satisfiability of $\widehat{\phi}_{\widehat{\mathcal{S}}}$. ❏

The set $\widehat{\mathcal{S}}$ can be computed in time exponential in the size of ϕ. In fact, there are exponentially many sets $X \subseteq \{p_1, \ldots, p_n\}$ to be considered, and testing consistency of $\mathcal{B}_X$ for each of these sets can be done in exponential time.

This leaves us with the problem of testing satisfiability of the propositional LTL formula

$$\widehat{\phi}_{\widehat{\mathcal{S}}} = \widehat{\phi} \wedge \Box \left(\bigvee_{X \in \widehat{\mathcal{S}}} \left(\bigwedge_{p \in X} p \wedge \bigwedge_{p \notin X} \neg p \right) \right)$$

in time exponential in the size of ϕ. Since the size of $\widehat{\phi}$ is bounded by the size of ϕ, it is sufficient to give an exponential upper bound in the size of $\widehat{\phi}$. To this purpose, note that the only effect of the box-formula in $\widehat{\phi}_{\widehat{\mathcal{S}}}$ is to restrict the worlds w in a propositional LTL structure satisfying $\widehat{\phi}$ to being induced by one of the elements of $\widehat{\mathcal{S}}$. Given a world w in a propositional LTL structure, we say that it is induced by a set $X \subseteq \{p_1, \ldots, p_n\}$ (written $w = w_X$) iff we have $p_i \in X$ iff w makes p_i true $(i = 1, \ldots, n)$.

Lemma 5.3. *The propositional LTL structure $\widehat{\mathfrak{I}} = (w_\iota)_{\iota=0,1,\ldots}$ satisfies $\widehat{\phi}_{\widehat{\mathcal{S}}}$ iff it satisfies $\widehat{\phi}$ and for every world w_ι of $\widehat{\mathfrak{I}}$ there is a set $X \in \widehat{\mathcal{S}}$ such that $w_\iota = w_X$.*

One way of deciding satisfiability of a propositional LTL formula $\widehat{\phi}$ is to construct a Büchi automaton $\mathcal{A}_{\widehat{\phi}}$ that accepts the propositional LTL structures satisfying $\widehat{\phi}$. To be more precise, let $\Sigma := \mathcal{P}(\{p_1, \ldots, p_n\})$. Then the propositional LTL structure $\widehat{\mathfrak{I}} = (w_\iota)_{\iota=0,1,\ldots}$ can be represented by an infinite word $X_0 X_1 \ldots$ over Σ, where X_ι is such that $w_\iota = w_{X_\iota}$. The Büchi automaton $\mathcal{A}_{\widehat{\phi}}$ is built such that it accepts exactly those infinite words over Σ that represent propositional LTL structures satisfying $\widehat{\phi}$. Consequently, $\widehat{\phi}$ is satisfiable iff the language accepted by $\mathcal{A}_{\widehat{\phi}}$ is non-empty. The size of $\mathcal{A}_{\widehat{\phi}}$ is exponential in the size of $\widehat{\phi}$, and the emptiness test for Büchi automata is polynomial in the size of the automaton.

Given such an automaton $\mathcal{A}_{\widehat{\phi}}$ for $\widehat{\phi}$ we can easily modify it into one accepting exactly the words representing propositional LTL structures satisfying $\widehat{\phi}_{\widehat{\mathcal{S}}}$. In fact, we just need to remove all transitions that use a letter from $\Sigma \setminus \widehat{\mathcal{S}}$. Obviously, this modification can be done in time polynomial in the size of $\mathcal{A}_{\widehat{\phi}}$, and thus in time exponential in the size of $\widehat{\phi}$. The size of the resulting automaton is obviously still only exponential in the size of $\widehat{\phi}$, and thus its emptiness can be tested in time exponential in the size of $\widehat{\phi}$. This yields the desired procedure that can check satisfiability of $\widehat{\phi}_{\widehat{\mathcal{S}}}$ in time exponential in the size of $\widehat{\phi}$. Overall, we have thus proved the ExpTime upper bound stated in the following theorem.

Theorem 5.4. *Satisfiability in $\mathcal{ALC}$-LTL without rigid names is an* ExpTime-*complete problem.*

Proof. We have already shown that the problem is in ExpTime. The hardness part of the theorem follows from the well-known fact that, in $\mathcal{ALC}$, satisfiability of a concept C w.r.t. a single GCI $C_1 \sqsubseteq C_2$ is ExpTime-complete [17]. Obviously, C is satisfiable w.r.t. $C_1 \sqsubseteq C_2$ iff the $\mathcal{ALC}$-LTL formula $a : C \wedge \Box(C_1 \sqsubseteq C_2)$ is satisfiable. ❑

Clearly, the ExpTime upper bound also holds for the restricted case with global GCIs. In addition, the formula $a : C \wedge \Box(C_1 \sqsubseteq C_2)$ constructed in the hardness part of the proof of Theorem 5.4 actually uses only global GCIs.

Corollary 5.5. *Satisfiability in $\mathcal{ALC}$-LTL$|_{gGCI}$ without rigid names is* ExpTime-*complete.*

6 Reasoning with rigid concepts

In this section, we consider the case where rigid concept names are allowed. First, note that, in contrast to temporal DLs where temporal operator may occur inside of concept descriptions, rigid concept names cannot easily be expressed within the logic without rigid concept names. In fact, the GCIs $A \sqsubseteq \Box A$ and $\neg A \sqsubseteq \Box\neg A$ express that A must be interpreted in a rigid way. However, they are not allowed by the syntax of $\mathcal{ALC}$-LTL since the box is applied directly to a concept, and not to an axiom.

We will show below that, for $\mathcal{ALC}$-LTL, the presence of rigid concept names indeed increases the complexity of the satisfiability problem, unless GCIs are restricted to being global. First, we treat the case of arbitrary GCIs, and then the special case of global GCIs.

Theorem 6.1. *Satisfiability in $\mathcal{ALC}$-LTL w.r.t. rigid concepts is* NExpTime-*complete.*

First, we show the lower bound.

Lemma 6.2. *Satisfiability in $\mathcal{ALC}$-LTL w.r.t. rigid concepts is* NExpTime-*hard.*

Proof. The proof is by reduction of a bounded version of the domino problem. A *domino system* is a triple $\mathfrak{D} = (D, H, V)$, where D is a finite set of *domino types* and $H, V \subseteq D \times D$ are the horizontal and vertical matching conditions. Let $\mathfrak{D}$ be a domino system and $I = d_0, \ldots, d_{n-1} \in D^*$ an *initial condition*, i.e. a sequence of domino types of length $n > 0$. A mapping $\tau : \{0, \ldots, 2^{n+1} - 1\} \times \{0, \ldots, 2^{n+1} - 1\} \to D$ is a 2^{n+1}*-bounded solution of* $\mathfrak{D}$ *respecting the initial condition* I iff, for all $x, y < 2^{n+1}$, the following holds:

- if $\tau(x, y) = d$ and $\tau(x \oplus_{2^{n+1}} 1, y) = d'$, then $(d, d') \in H$;
- if $\tau(x, y) = d$ and $\tau(x, y \oplus_{2^{n+1}} 1) = d'$, then $(d, d') \in V$;
- $\tau(i, 0) = d_i$ for $i < n$;

where $\oplus_{2^{n+1}}$ denotes addition modulo 2^{n+1}.

It is well-known [9, 22] that there is a domino system $\mathfrak{D} = (D, H, V)$ such that the following problem is NExpTime-hard: given an initial condition $I = d_0, \ldots, d_{n-1} \in D^*$, does $\mathfrak{D}$ have a 2^{n+1}-bounded solution respecting I or not?

We show that this problem can be reduced in polynomial time to satisfiability in $\mathcal{ALC}$-LTL w.r.t. rigid concepts. Interestingly, in our reduction we do not use any role names. All we need are the following concept and individual names:

- a single individual name a;
- the elements of D as *rigid* concept names, and a primed version of them as *non-rigid* concept names;
- *rigid* concept names $X_0, \ldots, X_n$ and $Y_0, \ldots, Y_n$ that are used to realize two binary counters modulo 2^{n+1}, where the X-counter expresses the horizontal and the Y-counter the vertical position of a domino;
- *non-rigid* concept names $Z_0, \ldots, Z_{2n+1}$ that are used to realize a binary counter modulo 2^{2n+2}, whose rôle will be explained below;
- an auxiliary *non-rigid* concept name N.

Intuitively, the first $n+1$ bits of the Z-counter are used to represent 2^{n+1} horizontal components $0 \leq x < 2^{n+1}$, and the second $n+1$ bits of the Z-counter are used to represent 2^{n+1} vertical components $0 \leq y < 2^{n+1}$. By counting with the Z-counter up to 2^{2n+2} in the temporal dimension, we ensure that every position $(x, y) \in \{0, \ldots, 2^{n+1} - 1\} \times \{0, \ldots, 2^{n+1} - 1\}$ is represented *in some world*. The counting is done using the individual name a, i.e., we enforce that, for every possible value of the Z-counter, there is a world where a belongs to the concepts from the corresponding subset of $\{Z_0, \ldots, Z_{2n+1}\}$. The rigid concept names $X_0, \ldots, X_n$ and $Y_0, \ldots, Y_n$ are then used to ensure that, *in every world*, there are individuals belonging to subsets of these concepts such that every position $(x, y) \in \{0, \dots, 2^{n+1} - 1\} \times \{0, \dots, 2^{n+1} - 1\}$ is realized in this world. Appropriate GCIs are used to ensure that (i) every position represented this way carries exactly one domino type; (ii) the horizontal and vertical matching conditions are respected; and (iii) the initial condition is satisfied.

Recall that we use $\phi \to \psi$ as an abbreviation for $\neg\phi \vee \psi$, $C \Rightarrow D$ as an abbreviation for $\neg C \sqcup D$, and $C \Leftrightarrow D$ as an abbreviation for $(C \Rightarrow D) \sqcap (D \Rightarrow C)$.

The reduction formula $\phi_{\mathcal{D},I}$ is the conjunction of the following formulae:

- for every possible value of the Z-counter, there is a world where a belongs to the concepts from the corresponding subset of $\{Z_0, \dots, Z_{2n+1}\}$:

$$\Box\big(\textstyle\bigwedge_{i \leq 2n+1} \big(\bigwedge_{j<i} a : Z_j \big) \to \big((a : Z_i \to \mathsf{X}(a : \neg Z_i)) \wedge (a : \neg Z_i \to \mathsf{X}(a : Z_i)) \big) \wedge \\ \textstyle\bigwedge_{i \leq 2n+1} \big(\bigvee_{j<i} a : \neg Z_j \big) \to \big((a : Z_i \to \mathsf{X}(a : Z_i)) \wedge (a : \neg Z_i \to \mathsf{X}(a : \neg Z_i)) \big) \big)$$

- the value of the Z-counter is shared by all individuals belonging to the current world: for all $i \leq 2n+1$

$$\Box \left((\top \sqsubseteq Z_i) \vee (\top \sqsubseteq \neg Z_i) \right)$$

- in every world, there is at least one individual for which the combined value of the X- and the Y-counter corresponds to the value of the Z-counter for a (and thus every individual) in this world:

$$\Box\big(\neg(\top \sqsubseteq \neg N) \wedge \\ \textstyle\bigwedge_{0 \leq i \leq n} (N \sqcap Z_i \sqsubseteq X_i) \wedge \bigwedge_{n+1 \leq i \leq 2n+1} (N \sqcap Z_i \sqsubseteq Y_{i-(n+1)}) \wedge \\ \textstyle\bigwedge_{0 \leq i \leq n} (N \sqcap \neg Z_i \sqsubseteq \neg X_i) \wedge \bigwedge_{n+1 \leq i \leq 2n+1} (N \sqcap \neg Z_i \sqsubseteq \neg Y_{i-(n+1)}) \big)$$

Since the concept names X_i, Y_i are rigid, this actually ensures that in every world every possible combination of values of the X- and Y-counters is realized by some individual. For a given such combination, the corresponding individual obviously must represent the same value combination in every world. Thus, for every position from $\{0, \dots, 2^{n+1}-1\} \times \{0, \dots, 2^{n+1}-1\}$ we have a world representing it with the help of the Z-counter, but we also have an individual representing it globally (i.e., in every world) with the help of the X- and Y-counters.

Having represented all positions from $\{0, \dots, 2^{n+1} - 1\} \times \{0, \dots, 2^{n+1} - 1\}$ in this way, we can now start to enforce an admissible tiling of these positions with domino types (i.e., a solution of the domino problem). As with the positions, we again have two copies of the tiling. One of them uses the primed concept names d' for $d \in D$ to tile the positions represented by the worlds with the help of the Z-counter. The other one uses the unprimed concept names $d \in D$ to tile the positions represented by the individuals with the help of the X- and Y-counters.

- every world gets exactly one domino type, expressed using the primed (and thus flexible) variant of the corresponding concept names:

$$\Box \left(\bigvee_{d \in D} (\top \sqsubseteq d') \quad \wedge \quad \bigwedge_{d,e \in D, d \neq e} (\top \sqsubseteq \neg(d' \sqcap e')) \right)$$

- the domino type of a given world is transfered globally to the individuals representing the same position as the world:

$$\Box \left(\bigsqcap_{0 \leq i \leq n} (Z_i \Leftrightarrow X_i) \sqcap \bigsqcap_{n+1 \leq i \leq 2n+1} (Z_i \Leftrightarrow Y_{i-(n+1)}) \quad \sqsubseteq \quad \bigsqcap_{d \in D} (d \Leftrightarrow d') \right)$$

Since the concept names d for $d \in D$ are rigid, this type is then associated with the individual in every world. Since every world has exactly one "primed" domino type (which is shared by all its individuals), every individual also has exactly one "unprimed" domino type: the one of the world representing the same position.

The two versions of the tiling can now be used to enforce the horizontal and vertical matching conditions. For example, the fact that the individual representing position $(x, y+1)$ is present in the world representing position (x, y) can be used to formulate the vertical matching condition.

We use the notation $C_X = C_Z^h$ ($C_Y = C_Z^v$) to express that the value of the X-counter agrees with the value represented by the first $n+1$ bits of the Z-counter (the value of the Y-counter agrees with the value represented by the second $n+1$ bits of the Z-counter). Accordingly, $(C_X = C_Z^h + 1 \bmod 2^{n+1})$ expresses that the value of the X-counter is equal to the value represented by the first $n+1$ bits of the Z-counter plus 1 (modulo 2^{n+1}). The intended meaning of the notation $(C_Y = C_Z^v + 1 \bmod 2^{n+1})$ should now be obvious. Details on how this can actually be expressed using concept descriptions are given in the proof of Lemma 4.2.

- the horizontal and vertical matching conditions are enforced as follows:

$$\begin{aligned} &\Box\big((C_X = C_Z^h) \sqcap (C_Y = C_Z^v + 1 \bmod 2^{n+1}) \quad \sqsubseteq \quad \bigsqcup_{(d,e) \in V} (d' \sqcap e)\big) \\ &\Box\big((C_Y = C_Z^v) \sqcap (C_X = C_Z^h + 1 \bmod 2^{n+1}) \quad \sqsubseteq \quad \bigsqcup_{(d,e) \in H} (d' \sqcap e)\big) \end{aligned}$$

For example, the first line looks at an individual that represents position $(x, y+1)$ in a world that represents position (x, y), and enforces that the domino type e associated with the individual (expressed by the rigid concept name e) is vertically compatible with the domino type d associated with the world (expressed by the flexible concept name d').

It remains to represent the initial condition $I = d_0, \ldots, d_{n-1}$. For this purpose, we employ the notation $C_Z^v = 0$ and $C_Z^h = i$ for $0 = 1, \ldots, n-1$, with the obvious meaning and the obvious representation by concept descriptions.

- for all $i = 0, \ldots, n-1$:

$$\Box\big((C_Z^v = 0) \sqcap (C_Z^h = i) \sqsubseteq d_i'\big)$$

This finishes the definition of the $\mathcal{ALC}$-LTL formula $\phi_{\mathfrak{D},I}$, which is the conjunction of the formulae introduced above. It is easy to see that the size of $\phi_{\mathfrak{D},I}$ is polynomial in n, and that $\phi_{\mathfrak{D},I}$ is satisfiable w.r.t. rigid concepts iff $\mathfrak{D}$ has a 2^{n+1}-bounded solution respecting I. ❏

Next, we show that the complexity lower bound provided by the above lemma is tight.

Lemma 6.3. *Satisfiability in $\mathcal{ALC}$-LTL w.r.t. rigid concepts is in* NExpTime.

Proof. We want to reuse the claim shown in the proof of Lemma 4.3:

Fact. *The $\mathcal{ALC}$-LTL formula ϕ is satisfiable w.r.t. rigid names iff there is a set $\mathcal{S} = \{X_1, \ldots, X_k\} \subseteq \mathcal{P}(\{p_1, \ldots, p_n\})$ such that the propositional LTL formula $\widehat{\phi}_{\mathcal{S}}$ is satisfiable and the Boolean $\mathcal{ALC}$-knowledge base $\mathcal{B} := \bigwedge_{1 \le i \le k} \mathcal{B}_i$ is consistent.*

If we apply this claim in the case where only concept names can be rigid, then we know that the Boolean $\mathcal{ALC}$-knowledge bases $\mathcal{B}_i$ are built over disjoint sets of role names. The only shared names are the rigid concept names. Assume we have guessed a set $\mathcal{S} = \{X_1, \ldots, X_k\} \subseteq \mathcal{P}(\{p_1, \ldots, p_n\})$, which can clearly be done within NExpTime. The formula $\widehat{\phi}_{\mathcal{S}}$ is itself of size exponential in the size of ϕ. However, we can use the same approach as in the proof of Theorem 5.4 to show that its satisfiability can actually be tested in time exponential in the size of ϕ.

Instead of testing the consistency of $\mathcal{B} = \bigwedge_{1 \le i \le k} \mathcal{B}_i$ directly (which would provide us with a double-exponential time bound), we try to reduce this test to k separate consistency tests, each requiring time exponential in the size of ϕ. Before we can do this, we need another guessing step. Assume that $A_1, \ldots, A_r$ are all the rigid concept names occurring in ϕ, and that $a_1, \ldots, a_s$ are all the individual names occurring in ϕ. We guess a set $\mathcal{T} \subseteq \mathcal{P}(\{A_1, \ldots, A_r\})$ and a mapping $\mathfrak{t} : \{a_1, \ldots, a_s\} \to \mathcal{T}$. Again, this guess can clearly be done within NExpTime.

Given $\mathcal{T}$ and $\mathfrak{t}$, we extend the knowledge bases $\mathcal{B}_i$ to knowledge bases $\widehat{\mathcal{B}}_i(\mathcal{T}, \mathfrak{t})$ as follows. For $Y \subseteq \{A_1, \ldots, A_r\}$, let C_Y be the concept description $C_Y := \bigsqcap\limits_{A \in Y} A \sqcap \bigsqcap\limits_{A \notin Y} \neg A$. We define

$$\widehat{\mathcal{B}}_i(\mathcal{T}, \mathfrak{t}) \;\; := \;\; \mathcal{B}_i \;\; \wedge \bigwedge_{\mathfrak{t}(a) = Y} a : C_Y \;\; \wedge \;\; \top \sqsubseteq \bigsqcup_{Y \in \mathcal{T}} C_Y \;\; \wedge \bigwedge_{Y \in \mathcal{T}} \neg(\top \sqsubseteq \neg C_Y)$$

Claim. *The Boolean $\mathcal{ALC}$-knowledge base $\mathcal{B} := \bigwedge_{1\leq i\leq k} \mathcal{B}_i$ is consistent iff there is a set $\mathcal{T} \subseteq \mathcal{P}(\{A_1,\ldots,A_r\})$ and a mapping $\mathfrak{t} : \{a_1,\ldots,a_s\} \to \mathcal{T}$ such that the Boolean knowledge bases $\widehat{\mathcal{B}}_i(\mathcal{T},\mathfrak{t})$ for $i = 1,\ldots,k$ are separately consistent.*

For the "only if" direction, assume that $\mathcal{B} = \bigwedge_{1\leq i\leq k} \mathcal{B}_i$ has a model $\mathcal{I} = (\Delta, \cdot^{\mathcal{I}})$. Let $\mathcal{T}$ consist of those sets $Y \subseteq \{A_1,\ldots,A_r\}$ such that there is a $d \in \Delta$ with $d \in (C_Y)^{\mathcal{I}}$, and let $\mathfrak{t}$ be the mapping satisfying $\mathfrak{t}(a) = Y$ iff $a^{\mathcal{I}} \in (C_Y)^{\mathcal{I}}$. It is easy to see that, with this choice of $\mathcal{T}$ and $\mathfrak{t}$, all the knowledge bases $\widehat{\mathcal{B}}_i(\mathcal{T},\mathfrak{t})$ for $i = 1,\ldots,k$ have $\mathcal{I}$ as model.

To show the "if" direction, assume that there is a set $\mathcal{T} \subseteq \mathcal{P}(\{A_1,\ldots,A_r\})$ and a mapping $\mathfrak{t} : \{a_1,\ldots,a_s\} \to \mathcal{T}$ such that the Boolean knowledge bases $\widehat{\mathcal{B}}_i(\mathcal{T},\mathfrak{t})$ for $i = 1,\ldots,k$ have models $\mathcal{I}_i = (\Delta_i, \cdot^{\mathcal{I}_i})$. We can assume without loss of generality[6] that

- the domains Δ_i are countably infinite, and
- in each model $\mathcal{I}_i$, the sets $Y \in \mathcal{T}$ are realized by countably infinitely many individuals, i.e., there are countably infinitely many elements $d \in \Delta_i$ such that $d \in (C_Y)^{\mathcal{I}}$.

Consequently, the domains Δ_i are partitioned into the countably infinite sets $\Delta_i(Y)$ (for $Y \in \mathcal{T}$), which are defined as follows:

$$\Delta_i(Y) := \{d \in \Delta_i \mid d \in (C_Y)^{\mathcal{I}_i}\}$$

In addition, for each individual name $a \in \{a_1,\ldots,a_s\}$ we have

$$a^{\mathcal{I}_i} \in \Delta_i(\mathfrak{t}(a))$$

We are now ready to define the model $\mathcal{I} = (\Delta, \cdot^{\mathcal{I}})$ of $\mathcal{B}$. As the domain of $\mathcal{I}$ we take the domain of $\mathcal{I}_1$, i.e., $\Delta := \Delta_1$. Accordingly, we define $\Delta(Y) := \Delta_1(Y)$ for all $Y \in \mathcal{T}$. Because of the properties stated above, there exist bijections $\pi_i : \Delta_i \to \Delta$ such that

- the restriction of π_i to $\Delta_i(Y)$ is a bijection between $\Delta_i(Y)$ and $\Delta(Y)$;
- π_i respects individual names, i.e., $\pi_i(a^{\mathcal{I}_i}) = a^{\mathcal{I}_1}$ holds for all $a \in \{a_1,\ldots,a_s\}$. (Note that we have the unique name assumption for individual names.)

We us these bijections to define the interpretation function $\cdot^{\mathcal{I}}$ of $\mathcal{I}$ as follows:

[6] This is an easy consequence of the fact that Boolean $\mathcal{ALC}$-knowledge bases always have a finite model and that the countably infinite disjoint union of a model of a Boolean $\mathcal{ALC}$-knowledge base is again a model of this knowledge base.

- If A is a flexible concept name, then $\mathcal{B}$ contains its copies $A^{(i)}$ for $i = 1, \ldots, k$. Their interpretation is defined as follows:

$$(A^{(i)})^{\mathcal{I}} := \{\pi(d) \mid d \in A^{\mathcal{I}_i}\}.$$

- All role names r are flexible, and $\mathcal{B}$ contains their copies $r^{(i)}$ for $i = 1, \ldots, k$. Their interpretation is defined as follows:

$$(r^{(i)})^{\mathcal{I}} := \{(\pi(d), \pi(e)) \mid (d, e) \in r^{\mathcal{I}_i}\}.$$

- If A is a rigid concept names, then we define

$$A^{\mathcal{I}} := A^{\mathcal{I}_1}$$

- If a is an individual name, then we define

$$a^{\mathcal{I}} := a^{\mathcal{I}_1}$$

To prove the claim, it remains to show that $\mathcal{I}$ is a model of all the knowledge bases $\mathcal{B}_i$ $(i = 1, \ldots, k)$. This is an immediate consequence of the fact that π_i is an isomorphism between $\mathcal{I}_i$ and $\mathcal{I}$ w.r.t. the concept and role names occurring in $\mathcal{B}_i$. The isomorphism condition is satisfied for flexible concepts and roles by our definition of $\cdot^{\mathcal{I}}$, and for individual names by our assumptions on π_i. Now, let A be a rigid concept name. We must show that $d \in A^{\mathcal{I}_i}$ iff $\pi_i(d) \in A^{\mathcal{I}}$ holds for all $d \in \Delta_i$. Since Δ_i is partitioned into the sets $\Delta_i(Y)$ for $Y \in \mathcal{T}$, we know that there is a $Y \in \mathcal{T}$ such that $d \in \Delta_i(Y)$, i.e., $d \in (C_Y)^{\mathcal{I}_i}$. In addition, we know that $\pi_i(d) \in \Delta(Y)$, i.e., $\pi_i(d) \in (C_Y)^{\mathcal{I}_1} = (C_Y)^{\mathcal{I}}$. This implies that $d \in A^{\mathcal{I}_i}$ iff $A \in Y$ iff $d \in A^{\mathcal{I}}$, which finishes the proof that π_i is an isomorphism between $\mathcal{I}_i$ and $\mathcal{I}$. Consequently, $\mathcal{I}$ is a model of $\mathcal{B} = \bigwedge_{1 \leq i \leq k} \mathcal{B}_i$, which in turn finishes the proof of the claim.

To finish the proof of the lemma, we show that consistency of the knowledge bases

$$\widehat{\mathcal{B}}_i(\mathcal{T}, \mathfrak{t}) \quad = \quad \mathcal{B}_i \quad \wedge \quad \bigwedge_{\mathfrak{t}(a)=Y} a : C_Y \quad \wedge \quad \top \sqsubseteq \bigsqcup_{Y \in \mathcal{T}} C_Y \quad \wedge \quad \bigwedge_{Y \in \mathcal{T}} \neg(\top \sqsubseteq \neg C_Y)$$

can be decided in time exponential in the size of the input formula ϕ. Note that this is not trivial. In fact, while the size of $\mathcal{B}_i \ \wedge \ \bigwedge_{\mathfrak{t}(a)=Y} a : C_Y$ is polynomial in the size of ϕ, the cardinality of $\mathcal{T}$, and thus the size of

$$\top \sqsubseteq \bigsqcup_{Y \in \mathcal{T}} C_Y \quad \wedge \quad \bigwedge_{Y \in \mathcal{T}} \neg(\top \sqsubseteq \neg C_Y)$$

can be exponential in the size of ϕ. Decidability of the consistency of $\widehat{\mathcal{B}}_i(\mathcal{T}, \mathfrak{t})$ in time exponential in the size of ϕ is, however, an immediate consequence of the next lemma.

Overall, this completes the proof of the current lemma. In fact, after two NExpTime guesses, all we have to do are k (i.e., exponentially many) ExpTime consistency tests. ❏

Let $\widehat{\mathcal{B}}$ be a Boolean $\mathcal{ALC}$-knowledge base of size n, $A_1, \ldots, A_r$ concept names occurring in $\widehat{\mathcal{B}}$, and $\mathcal{T} \subseteq \mathcal{P}(\{A_1, \ldots, A_r\})$. Note that this implies that the cardinality of $\mathcal{T}$ is at most exponential in n, and the size of each $Y \in \mathcal{T}$ is linear in n. We say that an interpretation $\mathcal{I} = (\Delta, \cdot^{\mathcal{I}})$ is a *model of* $\widehat{\mathcal{B}}$ *w.r.t.* $\mathcal{T}$ if it is a model of $\widehat{\mathcal{B}}$ that additionally satisfies

$$\mathcal{T} = \{Y \subseteq \{A_1, \ldots, A_r\} \mid \text{ there is } d \in \Delta \text{ such that } d \in (C_Y)^{\mathcal{I}}\}$$

Accordingly, we say that $\widehat{\mathcal{B}}$ *is consistent w.r.t.* $\mathcal{T}$ if $\widehat{\mathcal{B}}$ has a model w.r.t. $\mathcal{T}$. Obviously, $\widehat{\mathcal{B}}$ is consistent w.r.t. $\mathcal{T}$ iff the knowledge base

$$\widehat{\mathcal{B}} \quad \wedge \top \sqsubseteq \bigsqcup_{Y \in \mathcal{T}} C_Y \quad \wedge \quad \bigwedge_{Y \in \mathcal{T}} \neg(\top \sqsubseteq \neg C_Y)$$

is consistent.

Lemma 6.4. *Let $\widehat{\mathcal{B}}$ be a Boolean $\mathcal{ALC}$-knowledge base of size n, $A_1, \ldots, A_r$ concept names occurring in $\widehat{\mathcal{B}}$, and $\mathcal{T} \subseteq \mathcal{P}(\{A_1, \ldots, A_r\})$. Then, consistency of $\widehat{\mathcal{B}}$ w.r.t. $\mathcal{T}$ can be decided in time exponential in n.*

Proof. The proof is an adaptation of the proof of Theorem 2.27 in [12], which shows that the consistency problem for Boolean $\mathcal{ALC}$-knowledge bases is in ExpTime. For the sake of completeness, we describe this adaptation in detail.

In [12] it is assumed (without loss of generality) that $\mathcal{ALC}$-concept descriptions contain only the constructors $\sqcap$, $\neg$, and $\exists$, that all GCIs are of the form $\top \sqsubseteq C$, and that Boolean knowledge bases are built from such GCIs and concept and role assertions using only the connectives $\wedge$ and $\neg$. In the following, we assume that $\widehat{\mathcal{B}}$ satisfies these restrictions.

Let $ind(\widehat{\mathcal{B}})$ be the set of all individual names occurring in $\widehat{\mathcal{B}}$, and let $con(\widehat{\mathcal{B}})$ and $sub(\widehat{\mathcal{B}})$ respectively denote the closure under negation of the set of all concept descriptions (including subdescriptions) occurring in $\widehat{\mathcal{B}}$ and the set of all subformulae of $\widehat{\mathcal{B}}$. As usual, we identify $\neg\neg E$ with E. Thus, the cardinalities of the three sets introduced above are polynomial in n.

A *concept type for* $\widehat{\mathcal{B}}$ is a set $\mathbf{c} \subseteq con(\widehat{\mathcal{B}})$ such that

- $C \sqcap D \in \mathbf{c}$ iff $C, D \in \mathbf{c}$, for all $C \sqcap D \in con(\widehat{\mathcal{B}})$;
- $\neg C \in \mathbf{c}$ iff $C \notin \mathbf{c}$, for all $C \in con(\widehat{\mathcal{B}})$.

A *formula type for* $\widehat{\mathcal{B}}$ is a set $\mathbf{f} \subseteq sub(\widehat{\mathcal{B}})$ such that

- $\psi \wedge \chi \in \mathbf{f}$ iff $\psi, \chi \in \mathbf{f}$, for all $\psi \wedge \chi \in con(\widehat{\mathcal{B}})$;
- $\neg\psi \in \mathbf{f}$ iff $\psi \notin \mathbf{f}$, for all $\psi \in con(\widehat{\mathcal{B}})$.

Obviously, the number of concept and formula types is exponential in n.

A *model candidate for* $\widehat{\mathcal{B}}$ is a triple $(\mathcal{S}, \iota, \mathbf{f})$ such that $\mathcal{S}$ is a set of concept types for $\widehat{\mathcal{B}}$, $\iota : ind(\widehat{\mathcal{B}}) \to \mathcal{S}$ is a function, and $\mathbf{f}$ is a formula type for $\widehat{\mathcal{B}}$ such that

(a) $\widehat{\mathcal{B}} \in \mathbf{f}$;

(b) $a : C \in \mathbf{f}$ implies $C \in \iota(a)$;

(c) $(a, b) : r \in \mathbf{f}$ implies $\{\neg C \mid \neg\exists r.C \in \iota(a)\} \subseteq \iota(b)$.

The model candidate $(\mathcal{S}, \iota, \mathbf{f})$ for $\widehat{\mathcal{B}}$ is called a *quasimodel for* $\widehat{\mathcal{B}}$ if it additionally satisfies

(d) for every $\mathbf{c} \in \mathcal{S}$ and every $\exists r.C \in \mathbf{c}$, there is $\mathbf{d} \in \mathcal{S}$ such that $\{\neg D \mid \neg\exists r.D \in \mathbf{c}\} \cup \{C\} \subseteq \mathbf{d}$;

(e) for every $\mathbf{c} \in \mathcal{S}$ and every concept C, if $\neg C \in \mathbf{c}$, then $\top \sqsubseteq C \notin \mathbf{f}$;

(f) for every concept C, if $\neg(\top \sqsubseteq C) \in \mathbf{f}$, then there is a $\mathbf{c} \in \mathcal{S}$ such that $C \in \mathbf{c}$;

(g) $\mathcal{S}$ is not empty.

In [12] it is shown that $\widehat{\mathcal{B}}$ is consistent iff there is a quasimodel for $\widehat{\mathcal{B}}$. In order to characterize consistency of $\widehat{\mathcal{B}}$ w.r.t. $\mathcal{T}$, we need to add two additional conditions. The quasimodel $(\mathcal{S}, \iota, \mathbf{f})$ for $\widehat{\mathcal{B}}$ *respects* $\mathcal{T}$ if it additionally satisfies

(h) for every concept type $\mathbf{c} \in \mathcal{S}$, there is a set $Y \in \mathcal{T}$ such that $Y = \mathbf{c} \cap \{A_1, \ldots, A_r\}$;

(k) for every set $Y \in \mathcal{T}$, there is a concept type $\mathbf{c} \in \mathcal{S}$ such that $Y = \mathbf{c} \cap \{A_1, \ldots, A_r\}$.

A simple adaptation of the proof in [12] can be used to show that $\widehat{\mathcal{B}}$ is consistent w.r.t. $\mathcal{T}$ iff there is a quasimodel for $\widehat{\mathcal{B}}$ that respects $\mathcal{T}$.

Next, we show that the ExpTime-algorithm for checking the existence of a quasimodel described in [12] can be adapted to check for the existence of a quasimodel that respects $\mathcal{T}$. The adapted algorithm works as follows. Given $\widehat{\mathcal{B}}$ and $\mathcal{T}$, it enumerates all model candidates $(\mathcal{S}, \iota, \mathbf{f})$ for $\widehat{\mathcal{B}}$ where $\mathcal{S}$ is the set of *all* concept types for $\widehat{\mathcal{B}}$. Let $\mathfrak{C}_1, \ldots, \mathfrak{C}_N$ be these candidates. As shown in [12], there are at most exponentially many of these candidates and they can be enumerated in exponential time.

Set $i = 1$ and consider $\mathfrak{C}_i = (\mathcal{S}, \iota, \mathbf{f})$.

Step 1. Go through all concept types in $\mathcal{S}$. We call a concept type $\mathbf{c} \in \mathcal{S}$ *defective* if one of the following three conditions holds:

- (d) is violated for some concept $\exists r.C \in \mathbf{c}$;
- (e) is violated for some concept C with $\neg C \in \mathbf{c}$;
- (h) is violated.

If we have found a defective concept type $\mathbf{c}$, and this concept type is not in the range of ι, then we set $\mathcal{S} := \mathcal{S} \setminus \{\mathbf{c}\}$ and continue with Step 1 (i.e., again go through all concept types in $\mathcal{S}$ and check whether one of them is defective). If we have found a defective concept type $\mathbf{c}$ that is in the range of ι, then we stop considering $\mathfrak{C}_i$ and go to Step 3. If in some iteration of Step 1 we find that none of the concept types in $\mathcal{S}$ is defective, then we go to Step 2.

Step 2. Check whether the triple $(\mathcal{S}', \iota, \mathbf{f})$ obtained through the application of Step 1 satisfies (f), (g), and (k). If it does, then stop with output "quasimodel respecting $\mathcal{T}$ exists." Otherwise, go to Step 3.

Step 3. Set $i := i + 1$. If $i \leq N$, then go to Step 1. Otherwise, stop with output "no quasimodel respecting $\mathcal{T}$."

It is easy to see that this algorithm yields the output "quasimodel respecting $\mathcal{T}$ exists" iff $\widehat{\mathcal{B}}$ indeed has a quasimodel respecting $\mathcal{T}$.

It is also not hard to see that the algorithm runs in time exponential in n. The index i goes from 1 to N, where N is at most exponential in n. The cardinality of the set of all concept types is exponential in n, and in each iteration of Step 1 for a fixed index $i \in \{1, \ldots, N\}$, one defective concept type is removed (or Step 1 is terminated for this index i). Thus, for a fixed i, the number of iterations of Step 1 is at most exponential in n. Finally, every single iteration of Step 1 needs only exponential time. There are at most exponentially many concept types to be considered, and checking for a violation of (d), (e), or (h) can be done in exponential time. Note in particular that this is also true for (h): one just needs to go through the (exponentially many) elements of $\mathcal{T}$. Similarly, checking for a violation of (f), (g), or (k) in Step 2 can be done in exponential time. ❏

Restricting GCIs to global ones decreases the complexity of the satisfiability problem.

Theorem 6.5. *Satisfiability in $\mathcal{ALC}$-LTL$|_{gGCI}$ w.r.t. rigid concepts is* ExpTime-*complete.*

ExpTime-hardness is an immediate consequence of Corollary 5.5, which states that the problem is already ExpTime-hard without rigid concepts. Before proving the ExpTime upper bound, we introduce an additional notation. The conjunction of $\mathcal{ALC}$-axioms $\mathcal{B}$ is said to be *ϕ-exhaustive* if, for every individual name a and every rigid concept name A occurring in ϕ, either $a : A$ or $a : \neg A$ occurs as a conjunct in $\mathcal{B}$.

Lemma 6.6. *Satisfiability in $\mathcal{ALC}$-LTL$|_{gGCI}$ w.r.t. rigid concepts is in* ExpTime.

Proof. Consider an $\mathcal{ALC}$-LTL formula $\phi = \Box\mathcal{B} \wedge \varphi$, where $\mathcal{B}$ is a conjunction of $\mathcal{ALC}$-axioms and φ is an $\mathcal{ALC}$-LTL formula that does not contain GCIs. We can assume without loss of generality that $\mathcal{B}$ is ϕ-exhaustive. In fact, given an arbitrary Boolean $\mathcal{ALC}$-knowledge base $\mathcal{B}$, we can build all the ϕ-exhaustive knowledge bases $\mathcal{B}'$ that are obtained from $\mathcal{B}$ by conjoining to it, for every individual name a and every rigid concept name A occurring in ϕ, either $a : A$ or $a : \neg A$. Obviously, $\phi = \Box\mathcal{B} \wedge \varphi$ is satisfiable w.r.t. rigid concepts iff $\Box\mathcal{B}' \wedge \varphi$ is satisfiable w.r.t. rigid concepts for one of the extension $\mathcal{B}'$ of $\mathcal{B}$ obtained this way. Since the size of each such an extension is polynomial and there are only exponentially many such extensions, it is sufficient to show that testing satisfiability of $\Box\mathcal{B}' \wedge \varphi$ w.r.t. rigid concepts for ϕ-exhaustive knowledge bases $\mathcal{B}'$ is in ExpTime.

Thus, we assume in the following that $\mathcal{B}$ is ϕ-exhaustive. The proof that satisfiability of $\phi = \Box\mathcal{B} \wedge \varphi$ can be tested in ExpTime combines ideas used in the proofs of Lemma 4.3 and Theorem 5.4. Following the approach used in the proof of Lemma 4.3, we abstract every ABox assertion α_i occurring in φ by a propositional variable p_i, thus building the propositional LTL-formula $\widehat{\varphi}$. Similar to the proof of Theorem 5.4, we compute the set $\widehat{\mathcal{S}}$, which consists of those $X \subseteq \{p_1, \ldots, p_n\}$ for which the Boolean $\mathcal{ALC}$-knowledge base

$$\mathcal{B}_X \quad := \quad \mathcal{B} \wedge \bigwedge_{p_j \in X} \alpha_j \wedge \bigwedge_{p_j \notin X} \neg\alpha_j$$

is consistent. This computation can be done in exponential time since it requires exponentially many ExpTime consistency tests.

Claim. *Let $\phi = \Box\mathcal{B} \wedge \varphi$ be such that $\mathcal{B}$ is a ϕ-exhaustive conjunction of $\mathcal{ALC}$-axioms and φ is an $\mathcal{ALC}$-LTL formula not containing GCIs. Then ϕ is satisfiable w.r.t. rigid concepts iff the propositional LTL formula*

$$\widehat{\varphi}_{\widehat{\mathcal{S}}} \quad := \quad \widehat{\varphi} \wedge \Box\big(\bigvee_{X \in \widehat{\mathcal{S}}} \big(\bigwedge_{p_j \in X} p_j \wedge \bigwedge_{p_j \notin X} \neg p_j\big)\big)$$

is satisfiable.

Note that proving this claim actually completes the proof of our lemma. In fact, as shown in the proof of Theorem 5.4, satisfiability of $\widehat{\varphi}_{\widehat{\mathcal{S}}}$ can be decided in exponential time.

The proof of the "only if" direction of the claim is a simple combination of the arguments used in the proofs of (i) the "only if" direction of Lemma 5.2, and (ii) the "only if" direction of the claim shown in the proof of Lemma 4.3.

To show the "if" direction, assume that $\widehat{\varphi}_{\widehat{\mathcal{S}}}$ is satisfiable. Let $\widehat{\mathcal{I}} = (w_\iota)_{\iota=0,1,\dots}$ be a propositional LTL structure satisfying $\widehat{\varphi}_{\widehat{\mathcal{S}}}$. By construction, for every ι there is a set $X_\iota \subseteq \{p_1, \dots, p_n\}$ such that

- $w_\iota = w_{X_\iota}$, i.e., $X_\iota = \{p_j \mid j \in \{1, \dots, n\}$ and w_ι makes p_j true$\}$.
- the Boolean $\mathcal{ALC}$-knowledge base $\mathcal{B}_{X_\iota} = \mathcal{B} \wedge \bigwedge_{p_j \in X_\iota} \alpha_j \wedge \bigwedge_{p_j \notin X_\iota} \neg\alpha_j$ is consistent.

Let $\mathcal{I}_\iota = (\Delta^{\mathcal{I}_\iota}, \cdot^{\mathcal{I}_\iota})$ be a model of $\mathcal{B}_{X_\iota}$ ($\iota = 0, 1, \dots$). To complete the proof of the claim, we use the models $\mathcal{I}_\iota$ to construct models $\mathcal{J}_\iota = (\Delta, \cdot^{\mathcal{J}_\iota})$ of $\mathcal{B}_{X_\iota}$ that (i) have a common domain Δ, (ii) interpret the individual names in a rigid way, and (iii) respect rigid concept names. This, together with the fact that $\widehat{\mathcal{I}} = (w_\iota)_{\iota=0,1,\dots}$ satisfies $\widehat{\varphi}_{\widehat{\mathcal{S}}}$, then implies that the $\mathcal{ALC}$-LTL structure $(\mathcal{J}_\iota)_{\iota=0,1,\dots}$ satisfies ϕ and respects rigid concept names.

To simplify this construction, we assume that the models $\mathcal{I}_\iota$ have a certain restricted shape. In a DL interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$, we call an element $x \in \Delta^{\mathcal{I}}$ *named* if there is an individual name a such that $x = a^{\mathcal{I}}$; all other elements of $\Delta^{\mathcal{I}}$ are called *anonymous*. The following is easy to see:

Fact. *Any consistent Boolean $\mathcal{ALC}$-knowledge base has a model $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ that satisfies the following property for every role names r: if $(x, y) \in r^{\mathcal{I}}$ and x is anonymous, then y is also anonymous.*

For example, a standard tableau-based algorithm that test consistency of Boolean $\mathcal{ALC}$-knowledge base generates models that satisfy the property stated in the fact. In the following, we assume that the models $\mathcal{I}_\iota$ satisfy this property. We can also assume without loss of generality that the domains $\Delta^{\mathcal{I}_\iota}$ of these models are almost disjoint, except for the named individuals. To be more precises, let $\{a_1, \dots, a_\ell\}$ be the individual names occurring in ϕ, and and let $\Theta := \{x_1, \dots, x_\ell\}$ be a set of cardinality ℓ. We assume that $\Delta^{\mathcal{I}_\iota} \cap \Delta^{\mathcal{I}_{\iota'}} = \Theta$ for $\iota \neq \iota'$ and that $a_i^{\mathcal{I}_\iota} = x_i$ for $i = 1, \dots, \ell$.

Let us now build the new models $\mathcal{J}_\iota$ based on the models $\mathcal{I}_\iota$ satisfying these properties. The constant domain Δ is the union of the domains $\mathcal{I}_\iota$, i.e., Δ is obtained as the following disjoint union:

$$\Delta := \Theta \cup \bigcup_\iota (\Delta^{\mathcal{I}_\iota} \setminus \Theta).$$

The interpretation function $\cdot^{\mathcal{J}_\iota}$ of $\mathcal{J}_\iota$ is defined as follows:

- We define $a_i^{\mathcal{J}_\iota} := x_i$ for $i = 1, \dots, \ell$.

- If A is a concept name and $x \in \Delta^{\mathcal{J}_\kappa}$, we distinguish two cases, depending on whether x is named or anonymous:

  1. If x is named, then we define $x \in A^{\mathcal{J}_\iota}$ iff $x \in A^{\mathcal{I}_\iota}$.
  2. If x is anonymous, then we define $x \in A^{\mathcal{J}_\iota}$ iff $x \in A^{\mathcal{I}_\kappa}$.

- If r is a role name, $x \in \Delta^{\mathcal{J}_\kappa}$ and $y \in \Delta^{\mathcal{J}_\lambda}$, then we distinguish four cases:

  1. If x and y are named, then we define $(x, y) \in r^{\mathcal{J}_\iota}$ iff $(x, y) \in r^{\mathcal{I}_\iota}$.
  2. If x is named and y is anonymous, then we define $(x, y) \in r^{\mathcal{J}_\iota}$ iff $\iota = \lambda$ and $(x, y) \in r^{\mathcal{I}_\iota}$.
  3. If x and y are anonymous, then we define $(x, y) \in r^{\mathcal{J}_\iota}$ iff $\kappa = \lambda$ and $(x, y) \in r^{\mathcal{I}_\kappa}$.
  4. If x is anonymous and y is named, then $(x, y) \notin r^{\mathcal{J}_\iota}$.

By construction, the interpretations $\mathcal{J}_\iota$ have a common domain and they interpret the individual names in a rigid way. Next, we show that they respect rigid concept names, i.e., we have $A^{\mathcal{J}_{\iota_1}} = A^{\mathcal{J}_{\iota_2}}$ for all rigid concept names A and all $\iota_1, \iota_2 \geq 0$. Thus, assume that $x \in \Delta^{\mathcal{J}_\kappa}$ is given. We must show that $x \in A^{\mathcal{J}_{\iota_1}}$ iff $x \in A^{\mathcal{J}_{\iota_2}}$. If x is named, then we have $x = a^{\mathcal{I}_{\iota_1}} = a^{\mathcal{I}_{\iota_2}}$ for some individual name $a \in \{a_1, \ldots, a_\ell\}$. Since $\mathcal{B}$ was assumed to be ϕ-exhaustive, either $a : A$ or $a : \neg A$ occurs as a conjunct in $\mathcal{B}$. This, together with the fact that $\mathcal{I}_{\iota_1}$ and $\mathcal{I}_{\iota_2}$ are models of $\mathcal{B}$, implies that $x \in A^{\mathcal{I}_{\iota_1}}$ iff $x \in A^{\mathcal{I}_{\iota_2}}$, and thus $x \in A^{\mathcal{J}_{\iota_1}}$ iff $x \in A^{\mathcal{J}_{\iota_2}}$. If x is anonymous, then $x \in A^{\mathcal{I}_{\iota_1}}$ iff $x \in A^{\mathcal{I}_\kappa}$ iff $x \in A^{\mathcal{I}_{\iota_2}}$.

It remains to show that, for every $\iota \geq 0$, the interpretation $\mathcal{J}_\iota$ is a model of $\mathcal{B}_{X_\iota}$. The proof of this is based on the following two observations, in which C is assumed to be an arbitrary $\mathcal{ALC}$-concept description:

1. If x is named, then $x \in C^{\mathcal{J}_\iota}$ iff $x \in C^{\mathcal{I}_\iota}$.

2. If $x \in \Delta^{\mathcal{I}_\kappa}$ is anonymous, then $x \in C^{\mathcal{J}_\iota}$ iff $x \in C^{\mathcal{I}_\kappa}$.

Before proving these observations by induction on the structure of C, we show that they can indeed be used to prove that $\mathcal{J}_\iota$ is a model of $\mathcal{B}_{X_\iota}$.

First, assume that $C \sqsubseteq D$ is a GCI that occurs as a conjunct in $\mathcal{B}$. Consider an element $x \in C^{\mathcal{J}_\iota}$. We must show that $x \in D^{\mathcal{J}_\iota}$. If x is named, then $x \in C^{\mathcal{J}_\iota}$ implies $x \in C^{\mathcal{I}_\iota}$ by the above Observation 1. Since $\mathcal{I}_\iota$ is a model of $\mathcal{B}$, $x \in C^{\mathcal{I}_\iota}$ implies $x \in D^{\mathcal{I}_\iota}$, and thus $x \in D^{\mathcal{J}_\iota}$, again by Observation 1. Now, assume that x is anonymous, and that it belongs to $\Delta^{\mathcal{I}_\kappa}$ for some $\kappa \geq 0$. Then $x \in C^{\mathcal{J}_\iota}$ implies $x \in C^{\mathcal{I}_\kappa}$ by the above Observation 2. Since $\mathcal{I}_\kappa$ is a model of $\mathcal{B}$, $x \in C^{\mathcal{I}_\kappa}$ implies $x \in D^{\mathcal{I}_\kappa}$, and thus $x \in D^{\mathcal{J}_\iota}$, again by Observation 2.

Second, assume that $a : C$ is a concept assertion that occurs as a conjunct in $\mathcal{B}$ or in $\bigwedge_{p_j \in X_\iota} \alpha_j \wedge \bigwedge_{p_j \notin X_\iota} \neg\alpha_j$. (Note that negated concept assertions are also just assertions since $\neg(a : D)$ is equivalent to $a : \neg D$.) We know that $\mathcal{I}_\iota$ is a model of $\mathcal{B}_{X_\iota}$, and thus of $a : C$, i.e., $a^{\mathcal{I}_\iota} \in C^{\mathcal{I}_\iota}$. In addition, we have $a^{\mathcal{I}_\iota} = a^{\mathcal{J}_\iota}$, and since $a^{\mathcal{I}_\iota}$ is named, $a^{\mathcal{I}_\iota} \in C^{\mathcal{I}_\iota}$ iff $a^{\mathcal{I}_\iota} \in C^{\mathcal{J}_\iota}$. This shows that $\mathcal{J}_\iota$ is a model of the assertion $a : C$.

Third, assume that $(a, b) : r$ is a role assertion that occurs as a conjunct in $\mathcal{B}$ or in $\bigwedge_{p_j \in X_\iota} \alpha_j$. We know that $\mathcal{I}_\iota$ is a model of $\mathcal{B}_{X_\iota}$, and thus of $(a, b) : r$, i.e., $(a^{\mathcal{I}_\iota}, b^{\mathcal{I}_\iota}) \in r^{\mathcal{I}_\iota}$. In addition, we have $a^{\mathcal{I}_\iota} = a^{\mathcal{J}_\iota}$ and $b^{\mathcal{I}_\iota} = b^{\mathcal{J}_\iota}$. Since $a^{\mathcal{I}_\iota}, b^{\mathcal{I}_\iota}$ are named, $(a^{\mathcal{I}_\iota}, b^{\mathcal{I}_\iota}) \in r^{\mathcal{I}_\iota}$ implies $(a^{\mathcal{J}_\iota}, b^{\mathcal{J}_\iota}) \in r^{\mathcal{J}_\iota}$, by the definition of $r^{\mathcal{J}_\iota}$. Consequently, $\mathcal{J}_\iota$ is a model of $(a, b) : r$.

Finally, assume that $\neg((a, b) : r)$ is a negated role assertion that occurs as a conjunct in $\bigwedge_{p_j \notin X_\iota} \neg\alpha_j$. We know that $\mathcal{I}_\iota$ is a model of $\mathcal{B}_{X_\iota}$, and thus of $\neg((a, b) : r)$, i.e., $(a^{\mathcal{I}_\iota}, b^{\mathcal{I}_\iota}) \notin r^{\mathcal{I}_\iota}$. In addition, we have $a^{\mathcal{I}_\iota} = a^{\mathcal{J}_\iota}$ and $b^{\mathcal{I}_\iota} = b^{\mathcal{J}_\iota}$. Since $a^{\mathcal{I}_\iota}, b^{\mathcal{I}_\iota}$ are named, $(a^{\mathcal{I}_\iota}, b^{\mathcal{I}_\iota}) \notin r^{\mathcal{I}_\iota}$ implies $(a^{\mathcal{J}_\iota}, b^{\mathcal{J}_\iota}) \notin r^{\mathcal{J}_\iota}$, by the definition of $r^{\mathcal{J}_\iota}$. Consequently, $\mathcal{J}_\iota$ is a model of $\neg((a, b) : r)$. This finishes our proof that $\mathcal{J}_\iota$ is a model of $\mathcal{B}_{X_\iota}$.

It remains to show that the above observations are in fact correct. We prove this by induction on the structure of C. The base case (C is a concept name) is trivial by the definition of the interpretations $\mathcal{J}_\iota$. For the induction step, it is sufficient to consider conjunction, negation, and existential restrictions. Both conjunction and negation are trivial. Thus, the only interesting case is where C is of the form $\exists r.D$.

First, assume that x is named. If $x \in (\exists r.D)^{\mathcal{I}_\iota}$, then we know that there is an element $y \in \Delta^{\mathcal{I}_\iota}$ such that $(x, y) \in r^{\mathcal{I}_\iota}$ and $y \in D^{\mathcal{I}_\iota}$. If y is named, then this implies $(x, y) \in r^{\mathcal{J}_\iota}$ and $y \in D^{\mathcal{J}_\iota}$, where the first statement holds by the definition of $r^{\mathcal{J}_\iota}$ and the second by the induction hypothesis. This shows $x \in (\exists r.D)^{\mathcal{J}_\iota}$. Now, assume that y is anonymous. By our assumption that, except for the named individuals, the domains of the models $\mathcal{I}_\kappa$ are disjoint, we know that $(x, y) \in r^{\mathcal{I}_\iota}$ implies $y \in \Delta^{\mathcal{I}_\iota}$. As before, we obtain $(x, y) \in r^{\mathcal{J}_\iota}$ and $y \in D^{\mathcal{J}_\iota}$ by the definition of $r^{\mathcal{J}_\iota}$ and the induction hypothesis, respectively. This shows $x \in (\exists r.D)^{\mathcal{J}_\iota}$ also in this case.

Conversely, assume that x is named and $x \in (\exists r.D)^{\mathcal{J}_\iota}$. Then we know that there is an element $y \in \Delta$ such that $(x, y) \in r^{\mathcal{J}_\iota}$ and $y \in D^{\mathcal{J}_\iota}$. If y is named, then this implies $(x, y) \in r^{\mathcal{I}_\iota}$ and $y \in D^{\mathcal{I}_\iota}$, by the definition of $r^{\mathcal{J}_\iota}$ and the induction hypothesis, respectively. If y is anonymous, then the definition of $r^{\mathcal{J}_\iota}$ yields $y \in \Delta^{\mathcal{I}_\iota}$ and $(x, y) \in r^{\mathcal{I}_\iota}$. Since $y \in \Delta^{\mathcal{I}_\iota}$, $y \in D^{\mathcal{J}_\iota}$ implies $y \in D^{\mathcal{I}_\iota}$, by the induction hypothesis. Thus, we have $x \in (\exists r.D)^{\mathcal{I}_\iota}$ in both cases.

Second, assume that $x \in \Delta^{\mathcal{I}_\kappa}$ is anonymous. If $x \in (\exists r.D)^{\mathcal{I}_\kappa}$, then we know that there is an element $y \in \Delta^{\mathcal{I}_\kappa}$ such that $(x, y) \in r^{\mathcal{I}_\kappa}$ and $y \in D^{\mathcal{I}_\kappa}$. By our assumption on the models $\mathcal{I}_\kappa$, we know that y is also anonymous. The definition of $r^{\mathcal{J}_\kappa}$ thus yields $(x, y) \in r^{\mathcal{J}_\iota}$, and the induction hypothesis yields $y \in \Delta^{\mathcal{J}_\iota}$. Thus, we have $x \in (\exists r.D)^{\mathcal{J}_\iota}$.

Conversely, assume that $x \in \Delta^{\mathcal{I}_\kappa}$ is anonymous, and $x \in (\exists r.D)^{\mathcal{J}_\iota}$. Then we know that there is an element $y \in \Delta$ such that $(x, y) \in r^{\mathfrak{J}_\iota}$ and $y \in D^{\mathfrak{J}_\iota}$. The definition of $r^{\mathfrak{J}_\iota}$ implies that y is also anonymous, and that $y \in \Delta^{\mathcal{I}_\kappa}$ and $(x, y) \in r^{\mathcal{I}_\kappa}$. The induction hypothesis thus yields $y \in D^{\mathcal{I}_\kappa}$. This shows $x \in (\exists r.D)^{\mathcal{I}_\kappa}$. ❑

When defining the notion of an $\mathcal{ALC}$-LTL formula *with global GCIs*, we have restricted these formulae to being of the form $\phi = \Box\mathcal{B} \wedge \varphi$ where $\mathcal{B}$ is a conjunction of $\mathcal{ALC}$-axioms and φ is an $\mathcal{ALC}$-LTL formula that does not contain GCIs. Allowing also for negated $\mathcal{ALC}$-axioms as conjuncts in $\mathcal{B}$ would only add syntactic sugar, but would not increase the expressiveness of $\mathcal{ALC}$-LTL$|_{gGCI}$. In fact, a negated assertion $\neg\alpha$ occurring as a conjunct in $\mathcal{B}$ can be moved as a conjunct $\Box\neg\alpha$ to φ, and a negated GCI $\neg(C \sqsubseteq D)$ can be replaced by an assertion $a : C \sqcap \neg D$ in $\mathcal{B}$, where a is a new individual name.

One might think that it is even possible to relax the condition such that $\mathcal{B}$ is an arbitrary Boolean $\mathcal{ALC}$-knowledge base. This is, however, not the case since it would increase the complexity of the satisfiability problem to NExpTime.

Definition 6.7. *We say that ϕ is an $\mathcal{ALC}$-LTL formula* with global Boolean knowledge base *iff it is of the form $\phi = \Box\mathcal{B} \wedge \varphi$ where $\mathcal{B}$ is a Boolean $\mathcal{ALC}$-knowledge base and ψ is an $\mathcal{ALC}$-LTL formula that does not contain GCIs.*

A careful analysis of the proof of Lemma 6.2 shows that the $\mathcal{ALC}$-LTL formula constructed in the reduction is actually an $\mathcal{ALC}$-LTL formula with global Boolean knowledge base.

Corollary 6.8. *Satisfiability of $\mathcal{ALC}$-LTL w.r.t. rigid concepts and with global Boolean knowledge bases is* NExpTime*-complete.*

7 Restricting the temporal component

In this section, we consider the fragment $\mathcal{ALC}$-LTL$|_\Diamond$ of $\mathcal{ALC}$-LTL, in which $\Diamond$ is the only temporal operator. Our aim is to prove that satisfiability in $\mathcal{ALC}$-LTL$|_\Diamond$ w.r.t. rigid names is in ExpTime. The main reason for this is that we can restrict the attention to $\mathcal{ALC}$-LTL structures respecting rigid concept and role names that consist of only polynomially many distinct $\mathcal{ALC}$-interpretations. Before we can formulate this fact more formally in the next lemma,[7] we need to introduce some more notations. The *weight* of the $\mathcal{ALC}$-LTL structure $\mathfrak{I} = (\mathcal{I}_i)_{i=0,1,\ldots}$ is defined to be the cardinality of the set $\{\mathcal{I}_i \mid i = 0, 1, \ldots\}$.[8] The *set of subformulae* $\mathsf{sub}(\phi)$ of

[7]Note that this lemma is a straightforward generalization to $\mathcal{ALC}$-LTL$|_\Diamond$ of a very similar lemma for LTL$|_\Diamond$, the restriction of propositional LTL to its diamond fragment (see, e.g., Lemma 6.40 in [8]).

[8]Recall that all the $\mathcal{ALC}$-interpretations within one $\mathcal{ALC}$-LTL structure have the same domain. For this reason, we can use exact equality of interpretations rather than equality up to isomorphism when defining the weight of an $\mathcal{ALC}$-LTL structure.

the $\mathcal{ALC}$-LTL$|_\Diamond$ formula ϕ is defined in the obvious way, i.e., $\mathsf{sub}(\phi) = \{\phi\}$ if ϕ is an $\mathcal{ALC}$-axiom, $\mathsf{sub}(\Diamond\phi) = \{\Diamond\phi\} \cup \mathsf{sub}(\phi)$, $\mathsf{sub}(\phi \wedge \psi) = \{\phi \wedge \psi\} \cup \mathsf{sub}(\phi) \cup \mathsf{sub}(\psi)$, etc. The size of the $\mathcal{ALC}$-LTL$|_\Diamond$ formula ϕ is denoted by $|\phi|$.

Lemma 7.1. *If the $\mathcal{ALC}$-LTL$|_\Diamond$ formula ϕ is satisfiable w.r.t. rigid names, then there is an $\mathcal{ALC}$-LTL structure $\mathfrak{J}$ respecting rigid concept and role names of weight at most $|\phi| + 2$ such that $\mathfrak{J}, 0 \models \phi$.*

Proof. Let $\mathfrak{I}$ be an $\mathcal{ALC}$-LTL structure respecting rigid concept and role names such that $\mathfrak{I}, 0 \models \phi$. For each $\psi \in \mathsf{sub}(\phi)$ we use $P(\psi)$ to denote the set of time points at which $\mathfrak{I}$ makes ψ true, i.e.,

$$P(\psi) := \{i \mid i \geq 0 \text{ and } \mathfrak{I}, i \models \psi\}.$$

We claim that there is an $\ell > 0$ such that, for all $i \geq \ell$, we have that $\mathfrak{I}, i \models \psi$ implies that $P(\psi)$ is infinite. In fact, since $\mathsf{sub}(\phi)$ is finite, the set

$$P_{\text{fin}} := \bigcup_{\substack{\psi \in \mathsf{sub}(\phi) \\ P(\psi) \text{ finite}}} P(\psi)$$

is also finite. Thus, we can choose ℓ to be the least positive integer not belonging to P_{fin}.

For every $\psi \in sub(\phi)$, we choose a time point $p(\psi) \geq 0$ as follows:

- If $P(\psi)$ is finite, then $p(\psi)$ is the maximal element of $P(\psi)$, i.e., $p(\psi)$ is the maximal j with $\mathfrak{I}, j \models \psi$. Note that, in this case, we have $p(\psi) < \ell$ and $\mathfrak{I}, p(\psi) \models \psi$.
- If $P(\psi)$ is infinite, then $p(\psi)$ is an arbitrary time point $j \geq \ell$ such that $\mathfrak{I}, j \models \psi$. Note that, in this case, we have $p(\psi) \geq \ell$ and $\mathfrak{I}, p(\psi) \models \psi$.

Let $\mathsf{ran}(p)$ denote the range of the function $p : sub(\phi) \to \{0, 1, 2, \ldots\}$, and let $k_0, \ldots, k_{m-1}, k_m, \ldots, k_{n-1}$ be an enumeration of $\mathsf{ran}(p) \cup \{0, \ell\}$ such that

- $k_0 < \ldots < k_{m-1} < k_m < \ldots < k_{n-1}$ and
- $m \in \{1, \ldots, n-1\}$ is chosen such that $k_i < \ell$ iff $i < m$.

We define the $\mathcal{ALC}$-LTL structure $\mathfrak{J}$ as the sequence of $\mathcal{ALC}$-interpretations $\mathcal{I}_{k_0} \cdots \mathcal{I}_{k_{m-1}} (\mathcal{I}_{k_m} \cdots \mathcal{I}_{k_{n-1}})^\omega$, i.e., $\mathfrak{J} = (\mathcal{J}_i)_{i=0,1,\ldots}$ with $\mathcal{J}_i = \mathcal{I}_{f(i)}$, where

- $f(i) = k_i$ for $i < m$;
- $f(i) = k_{m+((i-m) \bmod (n-m))}$ for $i \geq m$.

Clearly, the fact that $\mathfrak{I}$ respects rigid concept names and role names implies that $\mathfrak{J}$ does the same. In addition, since the number of subformulae of ϕ is bounded by the size $|\phi|$ of ϕ, we have $n \leq |\phi| + 2$, and the weight of $\mathfrak{J}$ is bounded by n. Thus, it remains to show that $\mathfrak{J}, 0 \models \phi$. This is an immediate consequence of the following claim and the fact that $f(0) = k_0 = 0$ and $\phi \in \mathsf{sub}(\phi)$.

Claim. *For all $\psi \in \mathsf{sub}(\phi)$ and all $i \geq 0$, we have $\mathfrak{J}, i \models \psi$ iff $\mathfrak{I}, f(i) \models \psi$.*

The proof of the claim is by induction on the structure of ψ. We concentrate on the only non-trivial case, i.e., the case where ψ is of the form $\Diamond\chi$.

To show the "if" direction, assume that $\mathfrak{I}, f(i) \models \Diamond\chi$. First assume that $P(\chi)$ is finite. Let $s \geq 0$ be maximal such that $\mathfrak{I}, s \models \chi$. Since $\mathfrak{I}, f(i) \models \Diamond\chi$, we have $s \geq f(i)$. By the definition of the function p, we also have $p(\chi) = s$. In addition, by the definition of the function f, and since $s \geq f(i)$, there is a $j \geq i$ with $f(j) = s$. The induction hypothesis yields $\mathfrak{J}, j \models \chi$, and thus, by the semantics of the diamond operator, $\mathfrak{J}, i \models \Diamond\chi$.

Now assume that $P(\chi)$ is infinite. By the definition of the function p, we have $p(\chi) \geq \ell$. In addition, by the definition of the function f, there thus is a $j > i$ with $f(j) = p(\chi)$. Since $\mathcal{I}, p(\chi) \models \chi$, the induction hypothesis yields $\mathcal{J}, j \models \chi$, and thus $\mathcal{J}, i \models \Diamond\chi$. This completes the proof of the "if" direction.

To show the "only if" direction, assume that $\mathfrak{J}, i \models \Diamond\chi$. First, assume that $P(\chi)$ is finite. We know that there is a $j \geq i$ with $\mathfrak{J}, j \models \chi$. The induction hypothesis yields $\mathfrak{I}, f(j) \models \chi$. Since $P(\chi)$ is finite, we have $f(j) < \ell$, and the definition of f together with $j \geq i$ yields $f(j) \geq f(i)$. By the semantics of the diamond operator, this implies $\mathfrak{I}, f(i) \models \Diamond\chi$.

If $P(\chi)$ is infinite, then we have $\mathfrak{I}, s \models \Diamond\chi$ for every $s \geq 0$, and thus $\mathfrak{I}, f(i) \models \Diamond\chi$ is trivially satisfied. This completes the proof of the claim, and thus of the lemma. ❏

Given this lemma, we can now prove that satisfiability of $\mathcal{ALC}$-LTL$|_\Diamond$ formulae w.r.t. rigid names can be decided within deterministic exponential time.

Lemma 7.2. *Satisfiability in $\mathcal{ALC}$-LTL$|_\Diamond$ w.r.t. rigid names is in* ExpTime.

Proof. The proof of this lemma is very similar to the one of Lemma 4.3. We use the same notation as in that proof. The first step is to establish the following claim, where the set of propositional LTL$|_\Diamond$ formulae is defined in the obvious way, and where we use n to denote the number of $\mathcal{ALC}$-axioms occurring in ϕ.

Claim. *The $\mathcal{ALC}$-LTL$|_\Diamond$ formula ϕ is satisfiable w.r.t. rigid names iff there is a set $\mathcal{S} = \{X_1, \ldots, X_k\} \subseteq \mathcal{P}(\{p_1, \ldots, p_n\})$ of cardinality $k \leq |\phi| + 2$ such that the propositional LTL$|_\Diamond$ formula $\widehat{\phi}_{\mathcal{S}}$ is satisfiable and the Boolean $\mathcal{ALC}$-knowledge base $\mathcal{B} := \bigwedge_{1 \leq i \leq k} \mathcal{B}_i$ is consistent.*

The "if" direction of this claim is an immediate consequence of the corresponding claim shown in the proof of Lemma 4.3. For the "only if" direction, we can use the proof of the "only if" direction of the corresponding claim shown in the proof of Lemma 4.3. The only difference is that we start with an $\mathcal{ALC}$-LTL structure $\mathfrak{J}$ respecting rigid concept and role names *of weight at most* $|\phi| + 2$ such that $\mathfrak{J}, 0 \models \phi$. The existence of such a structure is guaranteed by Lemma 7.1. It is easy to see that then the set $\mathcal{S} = \{X_1, \ldots, X_k\}$ defined in the proof of Lemma 4.3 indeed is of cardinality $k \leq |\phi| + 2$.

This completes the proof of the claim. It remains to show that the claim provides us with a decision procedure for satisfiability in $\mathcal{ALC}$-LTL$|_\diamond$ w.r.t. rigid names that runs in deterministic exponential time. Let $m := |\phi|$. There are $\leq 2^{m(m+2)}$ subsets $\mathcal{S} \subseteq \mathcal{P}(\{p_1, \ldots, p_n\})$ of cardinality $\leq m + 2$ to be considered, and the size of each such subset $\mathcal{S} = \{X_1, \ldots, X_k\}$ is polynomial in m. Thus, the size of both $\widehat{\phi}_\mathcal{S}$ and $\mathcal{B} = \bigwedge_{1 \leq i \leq k} \mathcal{B}_i$ is polynomial in m. Since satisfiability in propositional LTL is in PSpace and the consistency problems for Boolean $\mathcal{ALC}$-knowledge bases is in ExpTime, this completes the proof of the lemma. ❑

ExpTime-hardness of satisfiability in $\mathcal{ALC}$-LTL$|_\diamond$ can be shown as in the proof of Theorem 5.4 by a reduction of the well-known ExpTime-hard problem of satisfiability of an $\mathcal{ALC}$-concept C w.r.t. a single GCI $C_1 \sqsubseteq C_2$. In fact, C is satisfiable w.r.t. $C_1 \sqsubseteq C_2$ iff the $\mathcal{ALC}$-LTL$|_\diamond$ formula $a : C \wedge \neg\Diamond\neg(C_1 \sqsubseteq C_2)$ is satisfiable w.r.t. rigid names.

Theorem 7.3. *Satisfiability in $\mathcal{ALC}$-LTL$|_\diamond$ w.r.t. rigid names is an* ExpTime-*complete problem.*

Obviously, the above reduction does not depend on the availability of rigid names, and the ExpTime decision procedure described in the proof of Lemma 7.2 also works in case there are only rigid concepts or no rigid names at all.

Corollary 7.4. *Satisfiability in $\mathcal{ALC}$-LTL$|_\diamond$ w.r.t. rigid concepts (without rigid names) is an* ExpTime-*complete problem.*

8 Conclusion

The faithful modeling of dynamically changing environments with a temporalized DL often requires the availability of rigid concepts and roles. We have shown that decidability and an elementary complexity upper bound can be achieved also in the presence of rigid roles by restricting the application of temporal operators to DL axioms. This is a big advance over the case where temporal operators can occur inside concept descriptions, in which rigid roles cause undecidability in the presence of a TBox and hardness for non-elementary time even without a TBox.

The decision procedures we have described in this paper were developed for the purpose of showing worst-case complexity upper bounds. The major topic for future work is to optimize them such that they can be used in practice, where we will first concentrate on the application scenario sketched in the introduction.

References

[1] A. Artale, R. Kontchakov, C. Lutz, F. Wolter, and M. Zakharyaschev. Temporalising tractable description logics. In *Proceedings of the Fourteenth International Symposium on Temporal Representation and Reasoning*. IEEE Computer Society Press, 2007.

[2] A. Artale, C. Lutz, and D. Toman. A description logic of change. In *Proc. of the 20th Int. Joint Conf. on Artificial Intelligence (IJCAI 2007)*, pages 218–223, 2007.

[3] Alessandro Artale and Enrico Franconi. A temporal description logic for reasoning about actions and plans. *J. of Artificial Intelligence Research*, 9:463–506, 1998.

[4] Alessandro Artale and Enrico Franconi. A survey of temporal extensions of description logics. *Ann. of Mathematics and Artificial Intelligence*, 30:171–210, 2000.

[5] Alessandro Artale and Enrico Franconi. Temporal description logics. In D. Gabbay, M. Fisher, and L. Vila, editors, *Handbook of Time and Temporal Reasoning in Artificial Intelligence*. The MIT Press, 2001.

[6] Alessandro Artale, Enrico Franconi, Frank Wolter, and Michael Zakharyaschev. A temporal description logic for reasoning over conceptual schemas and queries. In *Proc. of the 8th Eur. Workshop on Logics in Artificial Intelligence (JELIA'2002)"*, volume 2424 of *Lecture Notes in Computer Science*, pages 98–110. Springer-Verlag, 2002.

[7] Franz Baader, Diego Calvanese, Deborah McGuinness, Daniele Nardi, and Peter F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, 2003.

[8] Patrick Blackburn, Maarten de Rijke, and Yde Venema. *Modal Logic*, volume 53 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, 2001.

[9] Egon Börger, Erich Gräedel, and Yuri Gurevich. *The Classical Decision Problem.* Perspectives in Mathematical Logic. Springer-Verlag, 1997.

[10] Ashok K. Chandra, Dexter C. Kozen, and Larry J. Stockmeyer. Alternation. *J. of the ACM*, 28(1):114–133, 1981.

[11] Michael Finger and Dov Gabbay. Adding a temporal dimension to a logic system. *J. of Logic, Language and Information*, 2:203–233, 1992.

[12] Dov Gabbay, Agnes Kurusz, Frank Wolter, and M. Zakharyaschev. *Many-dimensional Modal Logics: Theory and Applications.* Elsevier, 2003.

[13] Ian Horrocks, Peter F. Patel-Schneider, and Frank van Harmelen. From SHIQ and RDF to OWL: The making of a web ontology language. *Journal of Web Semantics*, 1(1):7–26, 2003.

[14] Carsten Lutz. Interval-based temporal reasoning with general TBoxes. In *Proc. of the 17th Int. Joint Conf. on Artificial Intelligence (IJCAI 2001)*, pages 89–94, 2001.

[15] Carsten Lutz, Frank Wolter, and Michael Zakharyaschev. Temporal description logics: A survey. In *Proceedings of the Fifteenth International Symposium on Temporal Representation and Reasoning.* IEEE Computer Society Press, 2008.

[16] Amir Pnueli. The temporal logic of programs. In *Proc. of the 18th Annual Symp. on the Foundations of Computer Science (FOCS'77)*, pages 46–57, 1977.

[17] Klaus Schild. A correspondence theory for terminological logics: Preliminary report. In *Proc. of the 12th Int. Joint Conf. on Artificial Intelligence (IJCAI'91)*, pages 466–471, 1991.

[18] Klaus Schild. Combining terminological logics with tense logic. In *Proc. of the 6th Portuguese Conf. on Artificial Intelligence (EPIA'93)*, volume 727 of *Lecture Notes in Computer Science*, pages 105–120. Springer-Verlag, 1993.

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

[20] Albrecht Schmiedel. A temporal terminological logic. In *Proc. of the 8th Nat. Conf. on Artificial Intelligence (AAAI'90)*, pages 640–645, 1990.

[21] S. Schulz, K. Markó, and B. Suntisrivaraporn. Complex occurrents in clinical terminologies and their representation in a formal language. In *Proceedings of the First European Conference on SNOMED CT (SMCS'06)*, Copenhagen, Denmark, 2006.

[22] Stephan Tobies. A NEXPTIME-complete description logic strictly contained in C^2. In J. Flum and M. Rodríguez-Artalejo, editors, *Proc. of the Annual Conf. of the Eur. Assoc. for Computer Science Logic (CSL'99)*, volume 1683 of *Lecture Notes in Computer Science*, pages 292–306. Springer-Verlag, 1999.

[23] Frank Wolter and Michael Zakharyaschev. Temporalizing description logics. In M. de Rijke and D. Gabbay, editors, *Proc. of the 2nd Int. Workshop on Frontiers of Combining Systems (FroCoS'98)*, Amsterdam, 1999. Wiley.