ECAI 2010
H. Coelho et al. (Eds.)
IOS Press, 2010

doi:10.3233/978-1-60750-606-5-53

Verifying Properties of Infinite Sequences of Description Logic Actions

Franz Baader[1] and **Hongkai Liu**[1] and **Anees ul Mehdi**[2]

Abstract. The verification problem for action logic programs with non-terminating behaviour is in general undecidable. In this paper, we consider a restricted setting in which the problem becomes decidable. On the one hand, we abstract from the actual execution sequences of a non-terminating program by considering infinite sequences of actions defined by a Büchi automaton. On the other hand, we assume that the logic underlying our action formalism is a decidable description logic rather than full first-order predicate logic.

1 INTRODUCTION

Action programming languages like Golog [9] and Flux [13], which are respectively based on the situation calculus and the fluent calculus, can be used to control the behaviour of autonomous agents and mobile robots. Often, programs written in these languages are non-terminating since the robots are supposed to perform open ended tasks, like delivering coffee as long as there are requests. To ensure that the execution of such a program leads to the desired behaviour of the robot, one needs to specify the required properties in a formal way, and then verify that these requirements are met by any (infinite) execution of the program. In the coffee delivery example, one might, e.g., want to show that anyone requesting coffee will eventually get it delivered. When trying to automate this verification task, one has to deal with two sources of undecidability: (i) the expressiveness of the programming constructs (while loops, recursion) and (ii) the expressiveness of situation/fluent calculus, which encompasses full first-order predicate logic.

Verification for non-terminating Golog programs has first been addressed by De Giacomo, Ternovskaia, and Reiter [8], who express both the semantics of the programs and the properties to be verified using an appropriate fixpoint logic. To verify a property of a program, one first needs to compute a fixpoint, which is expressed in second-order logic. In general, this computation need not terminate (this corresponds to the first source of undecidability). Even if the fixpoint computation does terminate, verifying that the desired property holds requires a manual, meta-theoretic proof. Attempts to automate this approach are usually restricted to propositional logic [11]. Claßen and Lakemeyer [7] aim at the fully automated verification of non-terminating Golog programs. They specify properties in an extension of the situation calculus by constructs of the first-order temporal logic CTL*. Verification then basically boils down to the computation of a fixpoint, where again this computation need not terminate. If the fixpoint computation terminates, then the proof that the desired property holds is a deduction in the underlying logic (i.e., no meta-theoretic reasoning is required). However, due to the second source of undecidability mentioned above, this deduction problem is in general not decidable.

In the present paper, we introduce a restricted setting, where both sources of undecidability are avoided. Regarding the first source, instead of examining the actual execution sequences of a given Golog or Flux program, we consider infinite sequences of actions that are accepted by a given Büchi automaton $\mathcal{B}$. If $\mathcal{B}$ is an abstraction of the program, i.e. all possible execution sequences of the program are accepted by $\mathcal{B}$, then any property that holds in all the sequences accepted by $\mathcal{B}$ is also a property that is satisfied by any execution of the program. For example, assume that, among other actions, researcher John can perform the action "review paper," which makes him tired, and that robot Robin can perform the actions "deliver paper" and "deliver coffee," where the latter one results in John no longer being tired, whereas the former one results in John having to review yet another paper. The property ϕ_{tired} we want to ensure is that John does not stay tired indefinitely, i.e., whenever he is tired at some time point, then there is a later time point at which he is not tired. Assume that there is a complex program controlling Robin's behaviour, but we can show that Robin will infinitely often deliver coffee. Thus, the Büchi automaton $\mathcal{B}_{deliver}$ that accepts all action sequences that contain the action "deliver coffee" infinitely often is an abstraction of this program, and it is easy to see that any infinite sequence of actions accepted by this automaton satisfies ϕ_{tired}.

To avoid the second source of undecidability, we restrict the underlying logic to a decidable description logic. Description Logics (DLs) [2] are a well-known family of knowledge representation formalisms that may be viewed as fragments of first-order logic (FO). The main strength of DLs is that they offer considerable expressive power going far beyond propositional logic, while reasoning is still decidable. An action formalism based on DLs was first introduced in [5], and it was shown that important reasoning problems such as the projection problem, which are undecidable in the full situation/fluent calculus, are decidable in this restricted formalism.

In this paper, we show that these positive results can be extended to the verification problem. As logic for specifying properties of infinite sequences of DL actions, we use the temporalized DL $\mathcal{ALC}$-LTL recently introduced in [3], which extends the well-known propositional linear temporal logic (LTL) [12] by allowing for the use of axioms (i.e., TBox and ABox statements) of the basic DL $\mathcal{ALC}$ in place of propositional letters.[3] Note that the property ϕ_{tired} that we have used in the above coffee delivery example can easily be expressed in LTL.

In the next section, we first recall the basic definitions for DLs,

[1] TU Dresden, Germany, email: {baader,liu}@tcs.inf.tu-dresden.de; partially supported by DFG under grant BA 1122/10–2.
[2] KIT Karlsruhe, Germany, email: ame@aifb.uni-karlsruhe.de; partially supported by the EU in the project SOA4All (http://www.soa4all.eu).

[3] More precisely, we will consider the extension of $\mathcal{ALC}$-LTL to the more expressive DL $\mathcal{ALCO}$, but disallow TBox statements.

action formalisms based on DLs, temporalized DLs, and Büchi automata, and then introduce the verification problem and its dual, the satisfiability problem, which asks whether there is an infinite sequence of actions accepted by the given Büchi automaton $\mathcal{B}$ that satisfies the property. Since these problems are interreducible in polynomial time, we then concentrate on solving the satisfiability problem. In Section 3, we consider a restricted version of the general problem, where the Büchi automaton accepts exactly one infinite sequence of unconditional actions. The general problem is then investigated in Section 4. Because of space constraints, detailed proofs of our results and a formalization of the above example had to be omitted. They can be found in [4].

2 PRELIMINARIES

We start by introducing the DL $\mathcal{ALCO}$, which extends the basic DL $\mathcal{ALC}$ by nominals, i.e., singleton concepts.

Definition 1 *Let N_C, N_R, and N_I respectively be disjoint sets of* concept names*,* role names*, and* individual names*. The set of $\mathcal{ALCO}$-*concept descriptions *is the smallest set such that*

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

*An $\mathcal{ALCO}$-*concept definition *is of the form $A \equiv C$, where A is a concept name and C an $\mathcal{ALCO}$-concept description. An $\mathcal{ALCO}$-*TBox *$\mathcal{T}$ is a finite set of concept definitions with unique left-hand sides. Concept names occurring on the left-hand side of a definition of $\mathcal{T}$ are called* defined in *$\mathcal{T}$ whereas all other concept names are called* primitive in *$\mathcal{T}$. The TBox $\mathcal{T}$ is* acyclic *iff there are no cyclic dependencies between the definitions. An $\mathcal{ALCO}$-*ABox *$\mathcal{A}$ is a finite set of $\mathcal{ALCO}$-*assertions *of the form $C(a)$, $r(a,b)$, $\neg r(a,b)$, where C is an $\mathcal{ALCO}$-concept description, $r \in N_R$, and $a, b \in N_I$.*

We use $\top$ to abbreviate $A \sqcup \neg A$. Given an assertion γ, its negation $\neg\gamma$ is again an assertion: $\neg(C(a)) := (\neg C)(a)$, $\neg(r(a,b)) := \neg r(a,b)$, and $\neg(\neg r(a,b)) := r(a,b)$. We restrict our attention to acyclic TBoxes since, for more general TBox formalisms involving general concept inclusion axioms (GCIs), it is not clear how to define an appropriate semantics for DL actions [5]. The presence of nominals in the concept description language and of negated roles in ABoxes is needed for our treatment of DL actions (see [5]).

Definition 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 a set $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$ to every concept name A, a binary relation $r^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ to every role name r, and an element $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$ to every individual name a such that $a \neq b$ implies $a^{\mathcal{I}} \neq b^{\mathcal{I}}$ (UNA). This function is extended to $\mathcal{ALCO}$-concept descriptions as follows:*

- $\{a\}^{\mathcal{I}} = \{a^{\mathcal{I}}\}$;
- $(C \sqcap D)^{\mathcal{I}} = C^{\mathcal{I}} \cap D^{\mathcal{I}}, (C \sqcup D)^{\mathcal{I}} = C^{\mathcal{I}} \cup D^{\mathcal{I}}, (\neg C)^{\mathcal{I}} = \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$;
- $(\exists r.C)^{\mathcal{I}} = \{x \in \Delta^{\mathcal{I}} \mid \exists y \in \Delta^{\mathcal{I}}.\ (x,y) \in r^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\}$;
- $(\forall r.C)^{\mathcal{I}} = \{x \in \Delta^{\mathcal{I}} \mid \forall y \in \Delta^{\mathcal{I}}.\ (x,y) \in r^{\mathcal{I}} \rightarrow y \in C^{\mathcal{I}}\}$.

The interpretation $\mathcal{I}$ is a model *of the TBox $\mathcal{T}$ if $A^{\mathcal{I}} = C^{\mathcal{I}}$ for all $A \equiv C \in \mathcal{T}$, and of the ABox $\mathcal{A}$ if it* satisfies *all the assertions in $\mathcal{A}$, i.e., $a^{\mathcal{I}} \in C^{\mathcal{I}}$ for all $C(a) \in \mathcal{A}$, $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in r^{\mathcal{I}}$ for all $r(a,b) \in \mathcal{A}$, and $(a^{\mathcal{I}}, b^{\mathcal{I}}) \notin r^{\mathcal{I}}$ for all $\neg r(a,b) \in \mathcal{A}$. We say that $\mathcal{A}$ is* consistent w.r.t. *$\mathcal{T}$ if there is a model of $\mathcal{A}$ that is also a model of $\mathcal{T}$.*

The temporalized DL $\mathcal{ALCO}$-LTL is obtained from propositional linear temporal logic (LTL) [12] by allowing for the use of $\mathcal{ALCO}$-assertions in place of propositional letters (see [10] for a survey of temporalized DLs).

Definition 3 *$\mathcal{ALCO}$-LTL formulae are defined by induction:*

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

We use $\Box\phi$ to abbreviate $\neg(\top(a)\mathsf{U}\neg\phi)$. The difference to the logic $\mathcal{ALC}$-LTL introduced in [3] is, on the one hand, that $\mathcal{ALCO}$-assertions rather than just $\mathcal{ALC}$-assertions can be used. On the other hand, an $\mathcal{ALC}$-LTL formula may also contain GCIs, whereas in $\mathcal{ALCO}$-LTL we do not allow the use of terminological axioms. Instead, we use a global acyclic TBox, whose concept definitions must hold at every time point. The semantics of $\mathcal{ALCO}$-LTL is based on $\mathcal{ALCO}$-LTL structures, which are infinite sequences of interpretations over the same non-empty domain Δ in which every individual name stands for a unique element of Δ.

Definition 4 *An* $\mathcal{ALCO}$-LTL structure *is a sequence $\mathfrak{I} = (\mathcal{I}_i)_{i=0,1,\ldots}$ of $\mathcal{ALCO}$-interpretations $\mathcal{I}_i = (\Delta, \cdot^{\mathcal{I}_i})$ 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 $\mathcal{ALCO}$-LTL formula ϕ, an $\mathcal{ALCO}$-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:*

$\mathfrak{I}, i \models \beta$	*iff*	*$\mathcal{I}_i$ satisfies the $\mathcal{ALCO}$-assertion β*
$\mathfrak{I}, i \models \phi \wedge \psi$	*iff*	*$\mathfrak{I}, i \models \phi$ and $\mathfrak{I}, i \models \psi$*
$\mathfrak{I}, i \models \phi \vee \psi$	*iff*	*$\mathfrak{I}, i \models \phi$ or $\mathfrak{I}, i \models \psi$*
$\mathfrak{I}, i \models \neg\phi$	*iff*	*not $\mathfrak{I}, i \models \phi$*
$\mathfrak{I}, i \models \mathsf{X}\phi$	*iff*	*$\mathfrak{I}, i+1 \models \phi$*
$\mathfrak{I}, i \models \phi \mathsf{U} \psi$	*iff*	*there is $k \geq i$ such that $\mathfrak{I}, k \models \psi$ and $\mathfrak{I}, j \models \phi$ for all $j, i \leq j < k$*

In this paper, we assume that the transition from $\mathcal{I}_i$ to $\mathcal{I}_{i+1}$ in an $\mathcal{ALCO}$-LTL structure is caused by the application of an action. We recall the pertinent definitions for DL actions from [5]. For the sake of simplicity, we omit occlusions from our presentation.

Definition 5 *Let $\mathcal{T}$ be an acyclic $\mathcal{ALCO}$-TBox. An $\mathcal{ALCO}$-*action *α for $\mathcal{T}$ is a pair $(\mathsf{pre}, \mathsf{post})$ which consists of*

- *a finite set* pre *of $\mathcal{ALCO}$-assertions, the* pre-conditions*;*
- *a finite set* post *of* conditional post-conditions *of the form β/γ, where β is an $\mathcal{ALCO}$-assertion and γ is a* primitive literal for *$\mathcal{T}$, i.e., an assertion of the form $A(a)$, $\neg A(a)$, $r(a,b)$, or $\neg r(a,b)$ where A is a primitive concept name in $\mathcal{T}$, r is a role name, and a, b are individual names.*

If every $\beta/\gamma \in \mathsf{post}$ is of the form $\top(a)/\gamma$, then we call α an unconditional action*, and in this case we write γ instead of $\top(a)/\gamma$. Otherwise, it is a* conditional action*.*

Basically, such an action is applicable in an interpretation if its preconditions are satisfied. The conditional post-condition β/γ requires that γ must hold after the application of the action if β was satisfied before the application. In addition, nothing should change that is not required to change by some post-condition.

Definition 6 *Let $\mathcal{T}$ be an acyclic TBox, $\alpha = (\mathsf{pre}, \mathsf{post})$ an $\mathcal{ALCO}$-action for $\mathcal{T}$, and $\mathcal{I}, \mathcal{I}'$ interpretations sharing the same domain and*

interpreting all individual names in the same way. We say that α may transform $\mathcal{I}$ to $\mathcal{I}'$ w.r.t. $\mathcal{T}$ *(*$\mathcal{I} \Rightarrow_\alpha^\mathcal{T} \mathcal{I}'$*) if* $\mathcal{I}$ *and* $\mathcal{I}'$ *are both models of* $\mathcal{T}$ *and, for each primitive concept name* A *in* $\mathcal{T}$ *and each role name* r*, we have* $A^{\mathcal{I}'} = ((A^\mathcal{I} \cup A^+) \setminus A^-)$ *and* $r^{\mathcal{I}'} = ((r^\mathcal{I} \cup r^+) \setminus r^-)$*,*

$$\begin{array}{rrcl} \textit{where} & A^+ & = & \{b^\mathcal{I} \mid \beta/A(b) \in \mathsf{post} \wedge \mathcal{I} \models \beta\}, \\ & A^- & = & \{b^\mathcal{I} \mid \beta/\neg A(b) \in \mathsf{post} \wedge \mathcal{I} \models \beta\}, \\ & r^+ & = & \{(a^\mathcal{I}, b^\mathcal{I}) \mid \beta/r(a,b) \in \mathsf{post} \wedge \mathcal{I} \models \beta\}, \\ & r^- & = & \{(a^\mathcal{I}, b^\mathcal{I}) \mid \beta/\neg r(a,b) \in \mathsf{post} \wedge \mathcal{I} \models \beta\}. \end{array}$$

We say that α *is* executable *in* $\mathcal{I}$ *if* $\mathcal{I}$ *is a model of* pre*.*

It is an easy consequence of this definition that, for any model $\mathcal{I}$ of $\mathcal{T}$, there is exactly one model $\mathcal{I}'$ of $\mathcal{T}$ such that $\mathcal{I} \Rightarrow_\alpha^\mathcal{T} \mathcal{I}'$ [5]. An action that tries to add and remove the same literal at the same time does not really make sense. In the above definition, we have (arbitrarily) favoured removal of such a literal. However, in reality we just want to disallow such actions. For this reason, say that the action α is *consistent with* $\mathcal{T}$ if, for all $\beta_1/\gamma, \beta_2/\neg\gamma$ in the post-conditions of α, we have that the ABox $\{\beta_1, \beta_2\}$ is inconsistent w.r.t. $\mathcal{T}$. In the following we assume that all actions are consistent with $\mathcal{T}$.

We extend the notation $\Rightarrow_\alpha^\mathcal{T}$ to *finite* sequences of actions $u = \alpha_1 \cdots \alpha_m$ by writing $\mathcal{I} \Rightarrow_u^\mathcal{T} \mathcal{I}'$ if there are models $\mathcal{I}_1, \ldots, \mathcal{I}_{m-1}$ of $\mathcal{T}$ such that $\mathcal{I} \Rightarrow_{\alpha_1}^\mathcal{T} \mathcal{I}_1 \Rightarrow_{\alpha_2}^\mathcal{T} \mathcal{I}_2 \cdots \Rightarrow_{\alpha_{m-1}}^\mathcal{T} \mathcal{I}_{m-1} \Rightarrow_{\alpha_m}^\mathcal{T} \mathcal{I}'$. The *projection problem* is concerned with the question of whether a certain property holds after the execution of such a finite sequence of actions. Formally, this problem is defined as follows. Let $\mathcal{T}$ be an acyclic $\mathcal{ALCO}$-TBox, u a finite sequence of $\mathcal{ALCO}$-actions for $\mathcal{T}$, and $\mathcal{A}$ an $\mathcal{ALCO}$-ABox. The $\mathcal{ALCO}$-assertion β is a *consequence of applying* u *in* $\mathcal{A}$ *w.r.t.* $\mathcal{T}$ if, for all models $\mathcal{I}$ of $\mathcal{A}$ and $\mathcal{T}$ and all models $\mathcal{I}'$ of $\mathcal{T}$ with $\mathcal{I} \Rightarrow_u^\mathcal{T} \mathcal{I}'$, we have $\mathcal{I}' \models \beta$. As shown in [5], the projection problem for finite sequences of $\mathcal{ALCO}$-actions can be reduced to the consistency problem for $\mathcal{ALCO}$-ABoxes w.r.t. acyclic $\mathcal{ALCO}$-TBoxes (and vice versa), and thus is PSpace-complete. Note that this reduction crucially depends on the availability of nominals in the target language.

In this paper, we are interested in deciding whether the executions of *infinite* sequences of actions satisfy a (temporal) property expressed in $\mathcal{ALCO}$-LTL. Let Σ be a finite set of $\mathcal{ALCO}$-actions for $\mathcal{T}$. An *infinite sequence* of such actions can be viewed as an infinite word over the alphabet Σ, i.e., a mapping $w : \mathbb{N} \to \Sigma$, where $\mathbb{N}$ denotes the set of non-negative integers.

Definition 7 *Let* $\mathcal{T}$ *be an acyclic* $\mathcal{ALCO}$*-TBox,* $\mathcal{A}$ *an* $\mathcal{ALCO}$*-ABox, and* w *an infinite sequence of* $\mathcal{ALCO}$*-actions for* $\mathcal{T}$*. The* $\mathcal{ALCO}$*-LTL structure* $\mathfrak{I} = (\mathcal{I}_i)_{i=0,1,\ldots}$ *is* generated by w from $\mathcal{A}$ w.r.t. $\mathcal{T}$ *if* $\mathcal{I}_0$ *is a model of* $\mathcal{A}$ *and, for all* $i \geq 0$*, we have* $\mathcal{I}_i \Rightarrow_{w(i)}^\mathcal{T} \mathcal{I}_{i+1}$ *and* $w(i)$ *is executable in* $\mathcal{I}_i$*.*

For the verification problem, we do not examining the actual execution sequences of a given action program, but instead consider infinite sequences of actions that are accepted by a Büchi automaton abstracting such a program. *Büchi automata* are finite automata accepting infinite words [14]. A Büchi automaton $\mathcal{B}$ basically looks and works like a "normal" finite automaton, but it receives infinite words w as inputs, and thus generates infinite runs. An infinite run of $\mathcal{B}$ on w is an infinite word $r : \mathbb{N} \to Q$ over the alphabet Q of states of $\mathcal{B}$ such that $r(0)$ is an initial state and, for every $i \geq 0$, there is a transition of $\mathcal{B}$ from the state $r(i)$ with letter $w(i)$ to the state $r(i+1)$. This run is accepting if it infinitely often reaches a final state. The *language* $L_\omega(\mathcal{B})$ *of infinite words accepted by* $\mathcal{B}$ consists of all infinite words w over Σ such that $\mathcal{B}$ has an accepting run on w.

We are now ready to give a formal definition of the *verification problem*, which was informally introduced in Section 1, as the problem of deciding validity of an $\mathcal{ALCO}$-LTL formula w.r.t. an acyclic TBox, an ABox, and a Büchi automaton.

Definition 8 *Let* $\mathcal{T}$ *be an acyclic* $\mathcal{ALCO}$*-TBox,* $\mathcal{A}$ *an* $\mathcal{ALCO}$*-ABox,* Σ *a finite set of* $\mathcal{ALCO}$*-actions for* $\mathcal{T}$*,* $\mathcal{B}$ *a Büchi automaton for the alphabet* Σ*, and* ϕ *an* $\mathcal{ALCO}$*-LTL formula.*

- ϕ *is* valid w.r.t. $\mathcal{T}$, $\mathcal{A}$, and $\mathcal{B}$ *if* $\mathfrak{I}, 0 \models \phi$ *holds for all* $w \in L_\omega(\mathcal{B})$ *and all* $\mathcal{ALCO}$*-LTL structures* $\mathfrak{I}$ *generated by* w *from* $\mathcal{A}$ *w.r.t.* $\mathcal{T}$*.*
- ϕ *is* satisfiable w.r.t. $\mathcal{T}$, $\mathcal{A}$, and $\mathcal{B}$ *if there is* $w \in L_\omega(\mathcal{B})$ *and an* $\mathcal{ALCO}$*-LTL structure* $\mathfrak{I}$ *generated by* w *from* $\mathcal{A}$ *w.r.t.* $\mathcal{T}$ *such that* $\mathfrak{I}, 0 \models \phi$*.*

Obviously, ϕ is valid w.r.t. $\mathcal{T}$, $\mathcal{A}$ and $\mathcal{B}$ iff $\neg\phi$ is unsatisfiable w.r.t. $\mathcal{T}$, $\mathcal{A}$ and $\mathcal{B}$. For this reason, we concentrate in the following on solving the satisfiability problem.

3 THE CASE OF A SINGLE CYCLIC SEQUENCE OF UNCONDITIONAL ACTIONS

We say that the infinite word w is *cyclic* if it starts with an initial word $\alpha_1 \cdots \alpha_m$ and then repeats a non-empty word $\beta_1 \cdots \beta_n$ infinitely often. We denote such a cyclic word by $w = \alpha_1 \cdots \alpha_m (\beta_1 \cdots \beta_n)^\omega$. The following facts are well-known [14] (and easy to see): if $\mathcal{B}$ is a Büchi automaton that accepts a singleton language $\{w\}$, then w is a cyclic word of the form $w = \alpha_1 \cdots \alpha_m (\beta_1 \cdots \beta_n)^\omega$ where m, n are bounded by the cardinality of the set of states of $\mathcal{B}$; conversely any singleton language $\{w\}$ consisting of a cyclic word $w = \alpha_1 \cdots \alpha_m (\beta_1 \cdots \beta_n)^\omega$ is accepted by a corresponding Büchi automaton $\mathcal{B}_w$ such that the cardinality of the set of states of $\mathcal{B}$ is linear in $m + n$.

In this section, we consider only Büchi automata accepting singleton languages. In addition, we restrict the attention to unconditional actions. Thus, *for the remainder of this section, we assume that* $\mathcal{T}$ *is an acyclic* $\mathcal{ALCO}$*-TBox,* $\mathcal{A}$ *an* $\mathcal{ALCO}$*-ABox,* Σ *a finite set of* unconditional $\mathcal{ALCO}$*-actions for* $\mathcal{T}$*,* $\mathcal{B}_w$ *a Büchi automaton for the alphabet* Σ *accepting the singleton language* $\{w\}$ *for* $w = \alpha_1 \cdots \alpha_m (\beta_1 \cdots \beta_n)^\omega$*, and* ϕ *an* $\mathcal{ALCO}$*-LTL formula.* Such a cyclic sequence of actions represents a program that, after an initialization phase, runs in a non-terminating loop.

The main observation that allows us to solve the satisfiability problem for ϕ w.r.t. $\mathcal{T}$, $\mathcal{A}$ and $\mathcal{B}_w$ is that each $\mathcal{ALCO}$-LTL structure generated by w from $\mathcal{A}$ w.r.t. $\mathcal{T}$ "runs into a cycle" after the first $m + 2n$ interpretations.

Lemma 1 *Let* $\mathfrak{I} = (\mathcal{I}_i)_{i=0,1,\ldots}$ *be an* $\mathcal{ALCO}$*-LTL structure generated by* $w = \alpha_1 \cdots \alpha_m (\beta_1 \cdots \beta_n)^\omega$ *from* $\mathcal{A}$ *w.r.t.* $\mathcal{T}$*. Then* $\mathcal{I}_{m+kn+i} = \mathcal{I}_{m+n+i}$ *for all* $k \geq 2$ *and* $0 \leq i < n$*.*

Basically, we now apply the approach for solving the projection problem from [5] to the finite sequence of actions $\alpha_1 \cdots \alpha_m$ $\beta_1 \cdots \beta_n \beta_1 \cdots \beta_{n-1}$. In this approach, time-stamped copies of all concept and role names occurring in the input (i.e., in $w, \mathcal{T}, \mathcal{A}, \phi$) are generated, together with a number of additional auxiliary concept names. Using this extended vocabulary, one builds, for every assertion γ occurring in the input, time-stamped variants $\gamma^{(i)}$ for all $i, 0 \leq i \leq m + 2n - 1$. The extended vocabulary is also used to construct an acyclic $\mathcal{ALCO}$-TBox $\mathcal{T}_{\mathsf{red}}$ and an $\mathcal{ALCO}$-ABox $\mathcal{A}_{\mathsf{red}}$ such that the following holds:

- for every sequence $\mathcal{I}_0, \ldots, \mathcal{I}_{m+2n-1}$ of models of $\mathcal{T}$ such that $\mathcal{I}_0$ is a model of $\mathcal{A}$ and $\mathcal{I}_i \Rightarrow^{\mathcal{T}}_{w(i)} \mathcal{I}_{i+1}$ $(0 \leq i < m+2n-1)$, there is a model $\mathcal{J}$ of $\mathcal{A}_{\mathsf{red}}$ and $\mathcal{T}_{\mathsf{red}}$ such that

$$(*) \quad \mathcal{I}_i \text{ satisfies } \gamma \quad \text{iff} \quad \mathcal{J} \text{ satisfies } \gamma^{(i)}$$

  holds for all $i, 0 \leq i \leq m+2n-1$ and all assertions γ occurring in the input.
- for every model $\mathcal{J}$ of $\mathcal{A}_{\mathsf{red}}$ and $\mathcal{T}_{\mathsf{red}}$ there exists a sequence $\mathcal{I}_0, \ldots, \mathcal{I}_{m+2n-1}$ of models of $\mathcal{T}$ such that $\mathcal{I}_0$ is a model of $\mathcal{A}$, $\mathcal{I}_i \Rightarrow^{\mathcal{T}}_{w(i)} \mathcal{I}_{i+1}$ $(0 \leq i < m+2n-1)$, and $(*)$ holds for all $i, 0 \leq i \leq m+2n-1$ and all assertions γ occurring in the input.

By Lemma 1, any finite sequence $\mathcal{I}_0, \ldots, \mathcal{I}_{m+2n-1}$ satisfying the properties stated in the above items can be extended to an $\mathcal{ALCO}$-LTL structure generated by $w = \alpha_1 \cdots \alpha_m (\beta_1 \cdots \beta_n)^\omega$ from $\mathcal{A}$ w.r.t. $\mathcal{T}$ by setting $\mathcal{I}_{m+kn+i} := \mathcal{I}_{m+n+i}$ for all $k \geq 2$ and $0 \leq i < n$. We can enforce executability of the actions $w(j)$ in $\mathcal{I}_j$ by adding the ABox

$$\mathcal{A}_{\mathsf{pre}} = \bigcup_{0 \leq j < m+2n-1} \{\gamma^{(j)} \mid \gamma \in \mathsf{pre}_j\},$$

where pre_j is the set of pre-conditions of the action $w(j)$. To ensure that the $\mathcal{ALCO}$-LTL formula ϕ is satisfied, we generate an additional ABox $\mathcal{A}_\phi$ by applying a non-deterministic tableau algorithm. In this algorithm, we have time-stamped copies $\psi^{(i)}$ for every subformula ψ of ϕ. Note that, for the atomic subformulae (i.e., $\mathcal{ALCO}$-assertions), these are just the time-stamped copies introduced above. The tableau algorithm starts with the set $\mathcal{S} = \{\phi^{(0)}\}$ and then modifies this set by applying tableau rules. Instead of defining all these rules in detail, we just sketch the most interesting ones, which deal the temporal operators X and U (a complete description can be found in [4]).

There are two variants of the rule that deals with the X-operator. If, for some $i < m+2n-1$, we have $(\mathsf{X}\psi)^{(i)} \in \mathcal{S}$, then the first variant applies, which adds $\psi^{(i+1)}$ to $\mathcal{S}$ and removes $(\mathsf{X}\psi)^{(i)}$. If $(\mathsf{X}\psi)^{(m+2n-1)} \in \mathcal{S}$, then the second variant applies, which adds $\psi^{(m+n)}$ to $\mathcal{S}$ and removes $(\mathsf{X}\psi)^{(i)}$.

There are also two variants of the rule that deals with the U operator, depending on whether the until formula $(\psi_1 \mathsf{U} \psi_2)^{(i)} \in \mathcal{S}$ has a time-stamp $i \leq m+n$ or $i > m+n$. Here, we describe only the more interesting variant, which is the one for $i > m+n$. This variant (non-deterministically) picks a $k \in \{m+n, \ldots, m+2n-1\}$.

If $i \leq k \leq m+2n-1$, then the rule adds $\varphi_1^{(i)}, \ldots, \varphi_1^{(k-1)}, \varphi_2^{(k)}$ to $\mathcal{S}$ and removes $(\psi_1 \mathsf{U} \psi_2)^{(i)}$.

If $m+n \leq k < i$, then the rule adds $\varphi_1^{(i)}, \ldots, \varphi_1^{(m+2n-1)}$, $\varphi_1^{(m+n)}, \ldots, \varphi_1^{(k-1)}, \varphi_2^{(k)}$ and removes $(\psi_1 \mathsf{U} \psi_2)^{(i)}$.

It can be shown that rule application always terminates with a final set $\mathcal{S}$, which contains only (time-stamped) $\mathcal{ALCO}$-assertions, i.e., the final $\mathcal{S}$ is an ABox. Since the tableau algorithm has non-deterministic rules (such as the rule dealing with U), it can produce several such ABoxes, depending on the choices made in the non-deterministic rules. We say that the ABox $\mathcal{A}_\phi$ is *induced by ϕ w.r.t. w* if it is one of the ABoxes produced by applying the tableau algorithm to $\{\phi^{(0)}\}$.

In the restricted setting considered in this section, we can reduce the satisfiability problem introduced in Definition 8 to consistency of an $\mathcal{ALCO}$-ABox w.r.t. an acyclic $\mathcal{ALCO}$-TBox:

Lemma 2 *The $\mathcal{ALCO}$-LTL formula ϕ is satisfiable w.r.t. $\mathcal{T}$, $\mathcal{A}$, and $\mathcal{B}_w$ iff there is an ABox $\mathcal{A}_\phi$ induced by ϕ w.r.t. w such that $\mathcal{A}_{\mathsf{red}} \cup \mathcal{A}_{\mathsf{pre}} \cup \mathcal{A}_\phi$ is consistent w.r.t. $\mathcal{T}_{\mathsf{red}}$.*

The sizes of $\mathcal{A}_{\mathsf{red}}$, $\mathcal{A}_{\mathsf{pre}}$, and $\mathcal{T}_{\mathsf{red}}$ are polynomial in the size of $\mathcal{A}$, the size of $\mathcal{T}$, and w [5]. In addition, our tableau algorithm is an NPSpace-algorithm. Since NPSpace is equal to PSpace and the consistency problem for $\mathcal{ALCO}$-ABoxes w.r.t. acyclic $\mathcal{ALCO}$-TBoxes is in PSpace, this shows that we can decide the satisfiability problem within PSpace. PSpace-hardness follows from the fact that the PSpace-complete projection problem for $\mathcal{ALCO}$-actions can be reduced to the validity problem, which in turn can be reduced to the satisfiability problem. In [5], it is shown that the projection problem is PSpace-hard even for the empty TBox $\emptyset$, a fixed ABox $\mathcal{A}$, and a fixed unconditional action without preconditions α. It is easy to see that the assertion γ is a consequence of applying α in $\mathcal{A}$ w.r.t. $\emptyset$ iff the $\mathcal{ALCO}$-LTL formula $\mathsf{X}\gamma$ is valid w.r.t. $\emptyset$, $\mathcal{A}$ and $\mathcal{B}_w$ where $w = \alpha(\alpha)^\omega$.

Theorem 1 *Satisfiability and validity of an $\mathcal{ALCO}$-LTL formula w.r.t. an acyclic $\mathcal{ALCO}$-TBox, an $\mathcal{ALCO}$-ABox, and a Büchi automaton accepting a singleton language over an alphabet of unconditional actions are PSpace-complete.*

4 THE GENERAL CASE

Now, we consider arbitrary Büchi automata and (possibly) conditional actions. In this setting, we cannot use the approach introduced in the previous section. On the one hand, it is easy to see that, for conditional actions, the crucial Lemma 1 need not hold. On the other hand, while any non-empty language accepted by a Büchi automaton contains a cyclic word, it may also contain non-cyclic ones. Thus, it is not a priori clear whether a cyclic word can be taken as the word $w \in L_\omega(\mathcal{B})$ required by the definition of the satisfiability problem.

Our approach for solving satisfiability of an $\mathcal{ALCO}$-LTL formula ϕ w.r.t. an acyclic $\mathcal{ALCO}$-TBox $\mathcal{T}$, an $\mathcal{ALCO}$-ABox $\mathcal{A}$, and a Büchi automaton $\mathcal{B}$ over an alphabet Σ of (possibly) conditional actions is based on the approach for deciding satisfiability in $\mathcal{ALC}$-LTL introduced in [3]. Given an $\mathcal{ALC}$-LTL formula ϕ to be tested for satisfiability, this approach builds the *propositional abstraction* $\widehat{\phi}$ of ϕ by replacing each $\mathcal{ALC}$-assertion[4] γ occurring in ϕ by a corresponding propositional letter p_γ. Let $\mathcal{L}$ be the set of propositional letters used for the abstraction. Consider a set $\mathcal{S} \subseteq \mathcal{P}(\mathcal{L})$, i.e., a set of subsets of $\mathcal{L}$. Such a set induces the following (propositional) LTL formula:

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

Intuitively, this formula is satisfiable if there exists a propositional LTL structure satisfying $\widehat{\phi}$ in which, at every time point, the set of propositional letters satisfied at this time point is one of the sets $X \in \mathcal{S}$. To get satisfiability of ϕ from satisfiability of $\widehat{\phi}_{\mathcal{S}}$ for some $\mathcal{S} \subseteq \mathcal{P}(\mathcal{L})$, we must check whether the sets of assertions induced by the sets $X \in \mathcal{S}$ are consistent. To be more precise, assume that a set $\mathcal{S} = \{X_1, \ldots, X_k\} \subseteq \mathcal{P}(\mathcal{L})$ is given. For every $i, 1 \leq i \leq k$, and every concept name A (role name r) occurring in ϕ, we introduce a copy $A^{(i)}$ ($r^{(i)}$). We call $A^{(i)}$ ($r^{(i)}$) the ith copy of A (r). The assertion $\gamma^{(i)}$ is obtained from γ by replacing every occurrence of a concept or role name by its ith copy. The set $\mathcal{S} = \{X_1, \ldots, X_k\}$ induces the following ABox:

$$\mathcal{A}_{\mathcal{S}} := \bigcup_{1 \leq i \leq k} \{\gamma^{(i)} \mid p_\gamma \in X_i\} \cup \{\neg\gamma^{(i)} \mid p_\gamma \notin X_i\}.$$

[4] In [3], both assertions and GCIs need to be replaced. In the present paper, GCIs are not allowed to occur in LTL formulae, and thus we need to deal only with assertions.

The following lemma is proved in [3].

Lemma 3 *The $\mathcal{ALC}$-LTL formula ϕ is satisfiable iff there is a set $\mathcal{S} \subseteq \mathcal{P}(\mathcal{L})$ such that the propositional LTL formula $\widehat{\phi}_{\mathcal{S}}$ is satisfiable and the ABox $\mathcal{A}_{\mathcal{S}}$ is consistent (w.r.t. the empty TBox).*

Now, we show how we can use this approach to solve the satisfiability problem introduced in Definition 8, i.e., satisfiability of an $\mathcal{ALCO}$-LTL formula ϕ w.r.t. an acyclic $\mathcal{ALCO}$-TBox $\mathcal{T}$, an $\mathcal{ALCO}$-ABox $\mathcal{A}$, and a Büchi automaton $\mathcal{B}$ over an alphabet Σ of (possibly) conditional actions. First, note that Lemma 3 also holds if we formulate it for $\mathcal{ALCO}$-LTL formulae rather than $\mathcal{ALC}$-LTL formulae. However, the existence of a set $\mathcal{S} \subseteq \mathcal{P}(\mathcal{L})$ such that $\widehat{\phi}_{\mathcal{S}}$ is satisfiable and the ABox $\mathcal{A}_{\mathcal{S}}$ is consistent is not enough to have satisfiability of ϕ w.r.t. $\mathcal{T}$, $\mathcal{A}$, and $\mathcal{B}$. In fact, the existence of such a set only yields an $\mathcal{ALCO}$-LTL structure $\mathfrak{I} = (\mathcal{I}_i)_{i=0,1,\dots}$ satisfying ϕ. We also need to ensure (i) that $\mathcal{I}_0$ is a model of $\mathcal{A}$ and (ii) that there is an infinite word $w \in L_\omega(\mathcal{B})$ such that, for all $i \geq 0$, the transition from $\mathcal{I}_i$ to $\mathcal{I}_{i+1}$ is caused by the action $w(i)$ and $\mathcal{I}_i$ is a model of $\mathcal{T}$.

Ensuring that $\mathcal{I}_0$ is a model of $\mathcal{A}$ is easy since $\mathcal{A}$ can be encoded in the $\mathcal{ALCO}$-LTL formula by working with the formula $\phi \wedge \bigwedge_{\gamma \in \mathcal{A}} \gamma$ instead of ϕ. For this reason, we will assume in the following (without loss of generality) that *the ABox $\mathcal{A}$ is empty*.

To deal with the second issue, we introduce corresponding propositional letters p_γ not only for the assertions γ occurring in ϕ, but also for (i) the assertions γ occurring in the actions in Σ, and (ii) the assertions γ of the form $A(a)$ and $r(a, b)$ where A, r, a, b occur in ϕ, $\mathcal{T}$, or an action in Σ, A is a concept name that is primitive in $\mathcal{T}$, r is a role name, and a, b are individual names. We call the assertions introduced in (ii) *primitive assertions*. In the following, let $\mathcal{L}$ be the (finite) set of propositional letters obtained this way. Obviously, Lemma 3 still holds if we use this larger set of propositional letters to build the sets $\mathcal{S}$ and the formulae $\widehat{\phi}_{\mathcal{S}}$.

One way of deciding satisfiability of a propositional LTL formula $\widehat{\phi}$ is to construct a Büchi automaton $\mathcal{C}_{\widehat{\phi}}$ that accepts the propositional LTL structures satisfying $\widehat{\phi}$ [15]. To be more precise, let $\Gamma := \mathcal{P}(\mathcal{L})$. A propositional LTL structure $\widehat{\mathfrak{I}} = (w_i)_{i=0,1,\dots}$ is an infinite sequence of truth assignments to the propositional letters from $\mathcal{L}$. Such a structure can be represented by an infinite word $X = X(0)X(1)\dots$ over Γ, where $X(i)$ consists of the propositional variables that w_i makes true. The Büchi automaton $\mathcal{C}_{\widehat{\phi}}$ is built such that it accepts exactly those infinite words over Γ that represent propositional LTL structures satisfying $\widehat{\phi}$. Consequently, $\widehat{\phi}$ is satisfiable iff the language accepted by $\mathcal{C}_{\widehat{\phi}}$ is non-empty. The size of $\mathcal{C}_{\widehat{\phi}}$ is exponential in the size of $\widehat{\phi}$, and the emptiness test for Büchi automata is polynomial in the size of the automaton. As sketched in [3], the automaton $\mathcal{C}_{\widehat{\phi}}$ can easily be modified into one accepting exactly the words representing propositional LTL structures satisfying $\widehat{\phi}_{\mathcal{S}}$. In fact, we just need to remove all transitions that use a letter from $\Gamma \setminus \mathcal{S}$. Obviously, this modification can be done in time polynomial in the size of $\mathcal{C}_{\widehat{\phi}}$, and thus in time exponential in the size of $\widehat{\phi}$. We denote the Büchi automaton obtained this way by $\mathcal{C}^{\mathcal{S}}_{\widehat{\phi}}$.

Now, consider the Büchi automaton $\mathcal{B}$ from the input, and assume that it is of the form $\mathcal{B} = (Q, \Sigma, I, \Delta, F)$, where Q is the set of states, $I \subseteq Q$ the set of initial states, $\Delta \subseteq Q \times \Sigma \times Q$ the transition relation, and $F \subseteq Q$ the set of final states. We use $\mathcal{B}$ to construct a Büchi automaton $\mathcal{B}' = (Q', \Gamma, I', \Delta', F')$ that accepts those infinite words $X = X(0)X(1)\dots$ over the alphabet Γ for which there is an infinite word $w \in L_\omega(\mathcal{B})$ such that the difference between $X(i)$ and $X(i+1)$ is "caused by" the action $w(i)$:

- $\Gamma = \mathcal{P}(\mathcal{L})$;
- $Q' = Q \times \Sigma \times \Gamma$;
- $I' = I \times \Sigma \times \Gamma$;
- $((q, \alpha, X), Y, (q', \alpha', X')) \in \Delta'$ iff the following holds:
  1. $(q, \alpha, q') \in \Delta$;
  2. $X = Y$;
  3. Let $\alpha = (\mathsf{pre}, \mathsf{post})$.
     - $p_\gamma \in X$ for all $\gamma \in \mathsf{pre}$;
     - if $\beta/\gamma \in \mathsf{post}$ and $p_\beta \in X$ then $p_\gamma \in X'$;
     - for every primitive assertion γ, if $p_\gamma \in X$ and there is no $\beta/\neg\gamma \in \mathsf{post}$ with $p_\beta \in X$, then $p_\gamma \in X'$;
     - for every primitive assertion γ, if $p_\gamma \notin X$ and there is no $\beta/\gamma \in \mathsf{post}$ with $p_\beta \in X$, then $p_\gamma \notin X'$;
- $F' = F \times \Sigma \times \Gamma$.

The intersection of the languages $L_\omega(\mathcal{B}')$ and $L_\omega(\mathcal{C}^{\mathcal{S}}_{\widehat{\phi}})$ thus contains those infinite words $X = X(0)X(1)\dots$ over the alphabet Γ (i) that represent propositional LTL structures satisfying $\widehat{\phi}_{\mathcal{S}}$, and (ii) for which there is an infinite word $w \in L_\omega(\mathcal{B})$ such that the difference between $X(i)$ and $X(i+1)$ is caused by the action $w(i)$, where the formal meaning of "caused by" is given by the conditions in Item 3 of the definition of $\mathcal{B}'$. Since the class of languages of infinite words accepted by Büchi automata is closed under intersection, there is a Büchi automaton $\mathcal{D}(\widehat{\phi}, \mathcal{S}, \mathcal{B})$ accepting this intersection. This automaton can be obtained from $\mathcal{B}'$ and $\mathcal{C}^{\mathcal{S}}_{\widehat{\phi}}$ by a product construction that is a bit more complicated, but not more complex, than the construction for "normal" finite automata [14]. Thus, like $\mathcal{C}^{\mathcal{S}}_{\widehat{\phi}}$ and $\mathcal{B}'$, the automaton $\mathcal{D}(\widehat{\phi}, \mathcal{S}, \mathcal{B})$ is of size exponential in the size of the input.

Given a word $X = X(0)X(1)\dots$ accepted by $\mathcal{D}(\widehat{\phi}, \mathcal{S}, \mathcal{B})$, we still cannot be sure that the propositional LTL structure represented by this word can be lifted to an $\mathcal{ALCO}$-LTL structure generated by a word $w \in L_\omega(\mathcal{B})$ from the empty ABox w.r.t. $\mathcal{T}$. The first problem is that we must ensure that $X = X(0)X(1)\dots$ can be lifted to an $\mathcal{ALCO}$-LTL structure $\mathfrak{I} = (\mathcal{I}_i)_{i=0,1,\dots}$ satisfying ϕ. By Lemma 3, this is the case if the ABox $\mathcal{A}_{\mathcal{S}}$ is consistent (w.r.t. the empty TBox). However, we will see below that we need to adapt the definition of $\mathcal{A}_{\mathcal{S}}$ in order to align it with the approach used to solve the second problem.

This second problem is that we need to ensure that $\mathcal{I}_i \Rightarrow^{\mathcal{T}}_{w(i)} \mathcal{I}_{i+1}$ holds for all $i \geq 0$.[5] Note that Item 3 in the definition of $\mathcal{B}'$ only enforces that the changes to the *named part* of the interpretation (i.e., for the domain elements interpreting individual names) are according to the action $w(i)$. It does not say anything about the *unnamed part* of the interpretation (which, according to the semantics of our actions, should not be modified) and it does not deal with the TBox. Fortunately, this is exactly what the TBox $\mathcal{T}_{\mathsf{red}}$ already used in the previous section is designed for. Since we must align this TBox with the ABox $\mathcal{A}_{\mathcal{S}}$, we need to consider it in a bit more detail than it was necessary in Section 3. The idea is that every concept description C occurring in the input (directly or as subdescription) is represented by new concept names $T^{(i)}_C$ for $i = 1, \dots, k$, where the index i corresponds to the set $X_i \in \mathcal{S}$. In addition, we introduce copies $A^{(i)}, r^{(i)}$ $(i = 0, 1, \dots, k)$ for all concept and role names occurring in the input. Intuitively, for every index $i, 1 \leq i \leq k$, we want to have an interpretation $\mathcal{I}_i$ that is a model of the ABox

$$\mathcal{A}_i = \{\gamma \mid p_\gamma \in X_i\} \cup \{\neg\gamma \mid p_\gamma \notin X_i\}$$

[5] Recall that the definition of $\mathcal{I}_i \Rightarrow^{\mathcal{T}}_{w(i)} \mathcal{I}_{i+1}$ also includes the requirement that $\mathcal{I}_i$ must be a model of $\mathcal{T}$.

and of the input TBox $\mathcal{T}$, such that all these interpretations coincide on their unnamed parts. Now, for every concept name A (role name r), the copy $A^{(0)}$ ($r^{(0)}$) corresponds to the extension of A (r) on the unnamed part of $\mathcal{I}_i$ (which is the same for all i), and the copy $A^{(i)}$ ($r^{(i)}$) corresponds to the extension of A (r) on the named part of $\mathcal{I}_i$. For a concept description C, the concept name $T_C^{(i)}$ corresponds to the extension of C in $\mathcal{I}_i$ (both named and unnamed part). The TBox $\mathcal{T}_{\mathsf{red}}$ is defined such that, from a model of $\mathcal{T}_{\mathsf{red}}$, one can derive models $\mathcal{I}_i$ of $\mathcal{T}$ coinciding on their unnamed parts (see [5, 4] for details). To ensure that $\mathcal{I}_i$ is also a model of $\mathcal{A}_i$, we basically use the ABox $\mathcal{A}_\mathcal{S}$ introduced above. However, as ith copy $\widehat{\gamma}^{(i)}$ of a concept assertion $C(a)$ we now use $T_C^{(i)}(a)$ rather than the copy $\gamma^{(i)}$ used in [3] (see above). Let $\widehat{\mathcal{A}}_\mathcal{S}$ be defined like $\mathcal{A}_\mathcal{S}$, but with $\widehat{\gamma}^{(i)}$ replacing $\gamma^{(i)}$ for concept assertions. We are now ready to formlate the main technical result of this section (see [4] for the proof).

Lemma 4 *The $\mathcal{ALCO}$-LTL formula ϕ is satisfiable w.r.t. $\mathcal{T}$, $\emptyset$, and $\mathcal{B}$ iff there is a set $\mathcal{S} \subseteq \mathcal{P}(\mathcal{L})$ such that $L_\omega(\mathcal{D}(\widehat{\phi}, \mathcal{S}, \mathcal{B})) \neq \emptyset$ and $\widehat{\mathcal{A}}_\mathcal{S}$ is consistent w.r.t. $\mathcal{T}_{\mathsf{red}}$.*

This lemma yields an ExpSpace-decision procedure for the satisfiability problem. In fact, the double-exponentially many sets $\mathcal{S} \subseteq \mathcal{P}(\mathcal{L})$ can be enumerate within ExpSpace, and the exponentially large automaton $\mathcal{D}(\widehat{\phi}, \mathcal{S}, \mathcal{B})$ can be tested for emptiness in exponential time. Finally, the ABox $\widehat{\mathcal{A}}_\mathcal{S}$ is of exponential size (due to the fact that $\mathcal{S}$ is of exponential size) and the same is true for $\mathcal{T}_{\mathsf{red}}$. Since consistency w.r.t. an acyclic TBox is PSpace-complete in $\mathcal{ALCO}$, the required consistency test can be performed in ExpSpace.

Theorem 2 *Satisfiability and validity of an $\mathcal{ALCO}$-LTL formula w.r.t. an acyclic $\mathcal{ALCO}$-TBox, an $\mathcal{ALCO}$-ABox, and a Büchi automaton over an alphabet of (possibly) conditional actions are in ExpSpace.*

5 ADDITIONAL RESULTS AND RELATED AND FUTURE WORK

The results presented in this paper are not restricted to the DL $\mathcal{ALCO}$. In fact, just like the results in [5], they can be extended to all DLs between $\mathcal{ALC}$ and $\mathcal{ALCQIO}$. The approach basically stays the same, the main thing that changes is the complexity of the consistency problem for ABoxes w.r.t. acyclic TBoxes. For the restricted setting of Section 3, we can thus show that the satisfiability problem has the same complexity as the consistency problem for ABoxes w.r.t. acyclic TBoxes: it is PSpace-complete for the DLs $\mathcal{ALC}$, $\mathcal{ALCO}$, $\mathcal{ALCQ}$, and $\mathcal{ALCQO}$, ExpTime-complete for the DLs $\mathcal{ALCI}$ and $\mathcal{ALCIO}$, and NExpTime-complete for the DLs $\mathcal{ALCQI}$ and $\mathcal{ALCQIO}$. For the general setting considered in Section 4, we can show that the satisfiability problem is in ExpSpace for the DLs $\mathcal{ALC}$, $\mathcal{ALCO}$, $\mathcal{ALCQ}$, and $\mathcal{ALCQO}$, in 2-ExpTime for the DLs $\mathcal{ALCI}$ and $\mathcal{ALCIO}$, and in 2-NExpTime for the DLs $\mathcal{ALCQI}$ and $\mathcal{ALCQIO}$. The results for the general case also hold if actions are allowed to have occlusions. It is still an open problem whether the complexity upper-bounds for the general case are tight.

In [6], Calvanese et al. consider the problem of verifying action programs that perform ASK and TELL actions over DL-Lite ontologies. Though this work shares our general goal of verifying DL action programs, the technical set-up is quite different from ours: they use the inexpressive language DL-Lite rather than an expressive one like $\mathcal{ALCO}$, directly consider Golog programs rather than abstractions by automata, restrict the attention to finite sequence of actions, and do not employ a temporal logic for specifying properties.

In [1], $\mathcal{ALC}$-LTL is also used in the context of verification. The technical set-up and also the intended application context is, however, quite different. In [1] one observes changes to the world, without knowing how they are caused. Based on what one has observed so far, one tries to predict whether a property specified in $\mathcal{ALC}$-LTL can still be satisfied or falsified. In the present paper, we assume that we know the actions that cause changes, and that we have (an abstraction of) the control program that triggers the application of these actions. Based on this information, we try to verify a priori (before the program is executed) whether a property specified in $\mathcal{ALC}$-LTL is guaranteed to be satisfied.

In this paper, we have assumed that a Büchi automaton that abstracts a given action program in the sense that all possible execution sequences of the program are accepted by this automaton is already available. An important topic for future research is how to generate such an abstraction (semi)automatically from a given program.

ACKNOWLEDGEMENTS

We would like to thank Carsten Lutz, Giuseppe de Giacomo, and Gerhard Lakemeyer for helpful discussions.

REFERENCES

[1] F. Baader, A. Bauer, and M. Lippmann, 'Runtime verification using a temporal description logic', in *Proc. of FroCoS 2009*, pp. 149–164. Springer-Verlag.

[2] *The Description Logic Handbook: Theory, Implementation, and Applications*, eds., F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. F. Patel-Schneider, Cambridge University Press, 2003.

[3] F. Baader, S. Ghilardi, and C. Lutz, 'LTL over description logic axioms', in *Proc. of KR 2008*, pp. 684–694. AAAI Press.

[4] F. Baader, H. Liu, and A. ul Mehdi, 'Integrate Action Formalisms into Linear Temporal Description Logics', LTCS-Report 09-03, Institute for Theoretical Computer Science, TU Dresden, Germany, (2009). See http://lat.inf.tu-dresden.de/research/reports.html.

[5] F. Baader, C. Lutz, M. Miličić, U. Sattler, and F. Wolter, 'Integrating description logics and action formalisms: First results', in *Proc. of AAAI 2005*. AAAI Press. A long version of this paper, containing all technical details, was published as LTCS-Report 05-02, Institute for Theoretical Computer Science, TU Dresden, Germany. See http://lat.inf.tu-dresden.de/research/reports.html.

[6] D. Calvanese, G. De Giacomo, M. Lenzerini, and R. Rosati, 'Actions and programs over description logic ontologies', in *Proc. of DL 2007*.

[7] J. Claßen and G. Lakemeyer, 'A logic for non-terminating Golog programs', in *Proc. of KR 2008*, pp. 589–599. AAAI Press.

[8] G. De Giacomo, E. Ternovskaia, and R. Reiter, 'Non-terminating processes in the situation calculus', in *Proc. of the AAAI'97 Workshop on Robots, Softbots, Immobots: Theories of Action, Planning and Control*, (1997).

[9] H. J. Levesque, R. Reiter, Y. Lespérance, F. Lin, and R. B. Scherl, 'GOLOG: A logic programming language for dynamic domains', *J. of Logic Programming*, **31**(1–3), (1997).

[10] C. Lutz, F. Wolter, and M. Zakharyaschev, 'Temporal description logics: A survey', in *Proc. of TIME 2008*, pp. 3–14. IEEE Computer Society Press.

[11] N. Pelov and E. Ternovska, 'Reducing inductive definitions to propositional satisfiability', in *Proc. of ICLP 2005*, pp. 221–234. Springer.

[12] A. Pnueli, 'The temporal logic of programs', in *Proc. of FOCS 1977*, pp. 46–57. IEEE.

[13] M. Thielscher, 'FLUX: A logic programming method for reasoning agents', *Theory and Practice of Logic Programming*, **5**(4–5), pp. 533–565, (2005).

[14] W. Thomas, 'Automata on infinite objects', in *Handbook of Theoretical Computer Science*, volume B, 134–189, Elsevier, (1990).

[15] P. Wolper, M. Y. Vardi, and A. P. Sistla, 'Reasoning about infinite computation paths', in *Proc. of FOCS 1983*, pp. 185–194. IEEE.