The Hybrid μ-Calculus

Ulrike Sattler[1]* and Moshe Y. Vardi[2]**

[1] LuFG Theor. Informatik, RWTH Aachen, Germany, sattler@cs.rwth-aachen.de
[2] Department of Computer Science, Rice University, Houston, TX, vardi@rice.edu

Abstract. We present an ExpTime decision procedure for the full μ-Calculus (including converse programs) extended with nominals and a universal program, thus devising a new, highly expressive ExpTime logic. The decision procedure is based on tree automata, and makes explicit the problems caused by nominals and how to overcome them. Roughly speaking, we show how to reason in a logic lacking the tree model property using techniques for logics with the tree model property. The contribution of the paper is two-fold: we extend the family of ExpTime logics, and we present a technique to reason in the presence of nominals.

1 Introduction

Description Logics (DLs) are a family of knowledge representation formalisms designed for the representation of and reasoning about *terminological knowledge* [34, 28, 2]. Over the last years, they turned out to be also well-suited for the representation of and reasoning about, e.g., ontologies [31, 16] and database schemata, where they can support schema design, evolution, and query optimisation [7], source integration in heterogeneous databases/data warehouses [6], and conceptual modeling of multidimensional aggregation [18].

The basic notions of DLs are concepts (classes, unary predicates) and roles (binary predicates). A specific DL is mainly characterised by a set of constructors that allow to form complex concepts and roles from atomic ones. A standard DL knowledge base consists of two parts: in the *TBox*, the vocabulary of a given application domain is fixed. Some TBox formalisms only allow to introduce names for complex concepts, whereas others allow, additionally, to state general axioms such as $C \doteq D$ or $C \dot{\sqsubseteq} D$ for two (possibly complex) concepts [11, 22]. The second part of a DL knowledge base, the *ABox*, states facts concerning concrete individuals. Using the vocabulary fixed in the TBox, we can state in an ABox that the individual a is an instance of, e.g., the concept `CMReactor`, and that it is related via the role `has-part` to an individual b. Given such a "hybrid" knowledge base, interesting reasoning problems include the computation of the taxonomy (i.e., the hierarchy w.r.t. the subsumption relation) of those concepts defined in the TBox, finding inconsistent concepts defined in the TBox, and

* Part of this work was carried out while the second author was visiting Rice University on a DAAD Grant.
** Work partially supported by NSF grants CCR-9700061 and CCR-9988322

finding, for an individual a in the ABox, the most specific concepts defined in the TBox that a is an instance of.

To be of use in a specific application, a DL must provide the means to describe properties of objects that are relevant for this application. Unsurprisingly, the more expressive power a DL provides, the more complex the reasoning algorithms for this DL are. As a consequence, a variety of DLs were introduced together with investigations of the complexity of the corresponding reasoning algorithms/problems (see, e.g., [26, 34, 13]).

In 1991, Schild described the close relationship between DLs and modal logics or dynamic logics [32]. For example, it turned out that $\mathcal{ALC}$ is a notational variant of multi modal **K**. Following that, numerous new DLs with corresponding complexity results emerged by (extensions of) translations into modal and dynamic logics [9, 33, 10]. Due to its high expressive power, the full μ-calculus (i.e., propositional μ-calculus extended with converse programs) can be viewed as (one of) the "queens" of ExpTime modal/dynamic/temporal logics [23, 35, 40]. It is able to capture, for example, converse-PDL, CTL*, and other highly expressive modal/dynamic/temporal logics, and thus also highly expressive DLs [5]. Unfortunately, the μ-calculus lacks two features that are of great importance for it being also a "queen" for DLs: it does not provide an analogue for concept definition/general axioms that are provided by TBoxes, and it has no equivalent to ABox individuals. The first point is not a serious one since we could "internalise" general axioms using a greatest fixpoint formula even though the μ-calculus does not provide (constructors to build) a universal program [32]. The second one is more serious since, for example, the extension of the μ-calculus with individuals no longer has the tree model property. Moreover, in the presence of individuals, internalisation becomes more subtle.

In this paper, we extend the μ-calculus with a *universal role/program* to enable direct internalisation of TBoxes [32], and with a generalised form of ABox individuals, namely *nominals*, thus devising a logic where all standard inference problems concerning TBoxes and ABoxes can be reduced to satisfiability. In contrast to ABox individuals, nominals can be used inside complex formulae in the same place as atomic propositions. We are able to show that the complexity of the full μ-calculus, when extended with a universal program and nominals, does not increase, but remains in ExpTime. To prove this upper bound, we reduce satisfiability to the emptiness of alternating automata on infinite trees—a family of automata that can be viewed as abstractions of tableau algorithms. This technique is rather elegant in that it separates the logic from the algorithmics [39]. For example, a tableau-based algorithm might require sophisticated blocking techniques to guarantee termination [22]. Using the automata-theoretic technique, termination is not an issue since we can work on infinite trees. Moreover, this technique makes explicit which problems arise when reasoning in the presence of nominals and universal roles, and how to deal with them. We have chosen to deal with nominals by explicitly guessing most of the relevant information concerning nominals—a choice that will be explained in the sequel.

Besides being of interest by itself and, once again, showing the power of the automata-theoretic approach, the complexity result presented here broadens the range description/modal/dynamic logics that have ExpTime decision procedures. Over the last few years, it was shown that tableau-based algorithms for certain ExpTime-complete reasoning problems are amenable to optimisation and behave quite well in practise [21, 29, 19, 22]. Thus, establishing an ExpTime upper bound is a first step in developing a practical decision procedure for the hybrid μ-calculus or, at least, for fragments of this logic. We return to the practicality issue at the end of the paper.

Unfortunately, this new "queen" logic is still not "the queen" since it is missing a prominent feature, namely number restrictions/graded modalities [17, 12, 38]. This is due to the fact that, in the presence of converse roles and universal programs/roles (or any other means to internalise axioms), nominals and number restrictions/graded modalities lead to NExpTime-hardness [37].

From the tense logic perspective [4], the hybrid μ-calculus can also be viewed as one of the "queen" hybrid logics with ExpTime-complete reasoning problems: our result extends ExpTime-completeness results for, e.g., Priorean tense logic over transitive frames (which can be viewed as a notational variant of multi-modal $\mathbf{K}4$ with converse modalities) or converse-PDL with nominals in [1].

2 Preliminaries

In this section, we introduce syntax and semantics of the hybrid μ-calculus as well as two-way automata. It is the extension of the propositional μ-calculus with converse programs [40], a universal role, and nominals [30, 1], i.e., atomic formulae to refer to single points.

Definition 1. *Let* AP *be a set of atomic propositions,* Var *a set of propositional variables,* Nom *a set of nominals, and* Prog *a set of atomic programs with the universal program* $o \in \mathsf{Prog}$. *A* program *is either an atomic program or the converse* a^- *of an atomic program* $a \in \mathsf{Prog}$. *The set of formulae of the* hybrid μ-calculus *is the smallest set such that*

- **true**, **false**, p *and* $\neg p$ *are formulae for* $p \in \mathsf{AP} \cup \mathsf{Nom}$,
- $x \in \mathsf{Var}$ *is a formula,*
- *if* φ_1 *and* φ_2 *are formulae,* α *is a program, and* x *is a propositional variable, then* $\varphi_1 \wedge \varphi_2$, $\varphi_1 \vee \varphi_2$, $\langle\alpha\rangle\,\varphi_1$, $[\alpha]\,\varphi_1$, $\mu x.\varphi_1(x)$ *and* $\nu x.\varphi_1(x)$ *are formulae.*

A propositional variable $x \in \mathsf{Var}$ *is said to occur* free *in a formula if it occurs outside the scope of a fixpoint operator. A* sentence *is formula that contains no free propositional variable, i.e., each occurrence of a variable* x *is in the scope of a fixpoint operator* μ *or* ν. *We use* λ *to denote a fixpoint operator* μ *or* ν. *For a* λ*-formula* $\lambda x.\varphi(x)$, *we write* $\varphi(\lambda x.\varphi(x))$ *to denote the formula that is obtained by replacing each free occurrence of* x *in* φ *with* $\lambda x.\varphi(x)$.

Semantics is defined by means of a Kripke structure and, in the presence of variables and fixpoints, a valuation that associates a set of points with each

variable. Readers not familiar with fixpoints might want to look at [23, 35] for instructive examples and explanations of the semantics of the μ-calculus.

Definition 2. *Semantics of the hybrid μ-calculus is given by means of a* Kripke structure $K = (W, R, L)$, *where*

- W *is a set of points,*
- $R : \mathsf{Prog} \longrightarrow 2^{W \times W}$ *assigns to an atomic program a binary relation on* W,
- $R(o) = W \times W$, *and*
- $L : \mathsf{AP} \cup \mathsf{Nom} \longrightarrow 2^W$ *assigns to each atomic proposition or nominal the set of points in which it holds, such that* $L(n)$ *is a singleton for each nominal* n.

R *is extended to converse programs as follows:* $R(a^-) = \{(v, u) \mid (u, v) \in R(a)\}$.

Given a Kripke structure $K = (W, R, L)$ *and variables* $x_1, \ldots, x_m$, *a* valuation $\mathcal{V} : \{x_1, \ldots, x_m\} \longrightarrow 2^W$ *maps each variable to a subset of* W. *For a valuation* $\mathcal{V}$, *a variable* x, *and a set of points* $W' \subseteq W$, $\mathcal{V}[x/W']$ *is the valuation that is obtained from* $\mathcal{V}$ *by assigning* W' *to* x.

A formula φ *with free variables among* $x_1, \ldots, x_m$ *is interpreted over a Kripke structure* $K = (W, R, L)$ *as a mapping* φ^K *that associates, with each valuation* $\mathcal{V}$, *a subset* $\varphi^K(\mathcal{V})$ *of* W. *This mapping is defined inductively as follows:*

- $\mathbf{true}^K(\mathcal{V}) = W$, $\mathbf{false}^K(\mathcal{V}) = \emptyset$,
- *for* $p \in \mathsf{AP} \cup \mathsf{Nom}$, *we have* $p^K(\mathcal{V}) = L(p)$ *and* $(\neg p)^K(\mathcal{V}) = W \setminus L(p)$
- $(\varphi_1 \wedge \varphi_2)^K(\mathcal{V}) = (\varphi_1)^K(\mathcal{V}) \cap (\varphi_2)^K(\mathcal{V})$,
  $(\varphi_1 \vee \varphi_2)^K(\mathcal{V}) = (\varphi_1)^K(\mathcal{V}) \cup (\varphi_2)^K(\mathcal{V})$,
  $(\langle\alpha\rangle\, \varphi)^K(\mathcal{V}) = \{u \in W \mid$ *there is a* v *with* $(u, v) \in R(\alpha)$ *and* $v \in \varphi^K(\mathcal{V})\}$,
  $([\alpha]\, \varphi)^K(\mathcal{V}) = \{u \in W \mid$ *for all* v, $(u, v) \in R(\alpha)$ *implies* $v \in \varphi^K(\mathcal{V})\}$,
- $(\mu x.\varphi(x))^K(\mathcal{V}) = \bigcap\{W' \subseteq W \mid \varphi^K(\mathcal{V}[x/W']) \subseteq W'\}$
  $(\nu x.\varphi(x))^K(\mathcal{V}) = \bigcup\{W' \subseteq W \mid \varphi^K(\mathcal{V}[x/W']) \supseteq W'\}$

For a sentence ψ, *a Kripke structure* $K = (W, R, L)$, *and* $w \in W$, *we write* $K, w \models \psi$ *iff* $w \in \psi^K$, *and call* K *a* model *of* ψ.[1] *A sentence that has a model is called* satisfiable.

Remark 1. All formulae are by definition in negation normal form, i.e., negation occurs only in front of atomic propositions or nominals.

In the following, we will sometimes write $\psi(n_1, \ldots, n_\ell)$ to emphasize that $n_1, \ldots, n_\ell$ are exactly the nominals occurring in ψ.

Since we will treat atomic programs and their converse symmetrically, we will use $\overline{\alpha}$ to denote the converse of a program, i.e., a^- if $\alpha = a$ for some atomic program a, and b if $\alpha = b^-$ for some atomic program b. We use Prog_ψ to denote all (possibly negated) programs occurring in ψ.

In many decidable hybrid logics, we find formulae of the form $\varphi@n$ (to be read as " the formula φ holds at the nominal n") with the semantics

$$(\varphi@n)^K(\mathcal{V}) = \begin{cases} W & \text{if } n \in \varphi^K(\mathcal{V}) \\ \emptyset & \text{otherwise .} \end{cases}$$

[1] The interpretation of a sentence is independent of valuations.

We did not provide this operator since, in the presence of the universal role o, we can make use of the equivalence $\varphi@n \equiv [o](\neg n \vee \varphi)$.

We note that the formula $[o]n$ is satisfied only by a structure with a single state. This formula cannot be expressed without the use of both nominals and the universal program.

Finally, we introduce two-way alternating automata on infinite trees. This family of automata generalises non-deterministic tree automata in two ways: firstly, they allow for the rather elegant and succinct alternation [27], which allows for transitions such as "being in state q and seeing letter σ, the automaton either has an accepting run with q_1 from the left successor *and* an accepting run with q_2 from the right successor, or it has an accepting run with q' from the left successor." To express this kind of transitions, the transition functions involves positive boolean formulae instead of, e.g., sets of tuples of states as for non-deterministic automata. Secondly, being two-way allows runs to go up and down the input tree, similar to converse programs, which allow following programs in both directions. When running on a k-ary tree, a two-way automaton can have transitions going to the ith child and switching to state q' (denoted (i, q') with $1 \leq i \leq k$), staying at the same node switching to state q' (denoted $(0, q')$), or going to its (unique) predecessor and switching to state q' (denoted $(-1, q')$). For an introduction to two-way alternating automata and their application to the full μ-calculus, see [40].

Definition 3. *For $k \geq 1$ an integer, $(\{1, \ldots, k\}^*, V)$ is a* k-ary Σ-labelled tree *if V is a mapping that associates, with each node $x \in \{1, \ldots, k\}^*$, its label $V(x) \in \Sigma$. Intuitively, for $1 \leq i \leq k$, $x \cdot i$ is the ith child of x.*

Let $\mathcal{B}^+(X)$ be the set of positive Boolean formulae *(i.e., formulae built using $\wedge$ and $\vee$ only) over the set X. For $X' \subseteq X$, we say that X'* satisfies *a formula $\Theta \in \mathcal{B}^+(X)$ iff assigning* **true** *to all elements in X' and* **false** *to all elements in $X \setminus X'$ makes Θ true.*

Let $[k] = \{-1, 0, 1, \ldots, k\}$. A two-way alternating automaton *on k-ary Σ-labelled trees is a tuple $\mathbf{A} = (\Sigma, Q, \delta, q_0, F)$, where Q is a finite set of states, $q_0 \in Q$ is the initial state, $\delta : Q \times \Sigma \to \mathcal{B}^+([k] \times Q)$ is the transition relation, and F is the acceptance condition.*

A run *of $\mathbf{A}$ on a Σ-labelled k-ary tree (T, V) is a $(T \times Q)$-labelled tree (T_r, r) that satisfies the following conditions:*

- *$\epsilon \in T_r$ and $r(\epsilon) = (\epsilon, q_0)$,*
- *If $y \in T_r$ with $r(y) = (x, q)$ and $\delta(q, V(x)) = \Theta$, then there is a (possibly empty) set $S \subseteq [k] \times Q$ that satisfies Θ such that, for each $(c, q') \in S$, there is a node $y \cdot i \in T_r$ satisfying the following conditions:*
  - *If $c = \epsilon$, then $r(y \cdot i) = (x, q')$.*
  - *If $c \geq 1$, then $r(y \cdot i) = (x \cdot c, q')$.*
  - *If $c = -1$, then $x = x' \cdot i$ for some $1 \leq i \leq k$, and $r(y \cdot i) = (x', q')$.*

A run (T_r, r) is accepting *iff all its infinite paths satisfy the acceptance condition. Since we use tree automata for the μ-calculus, we consider the* parity *condition*

[36]. *A parity condition is given by an ascending chain of states of sets* $F = (F_0, \ldots, F_k)$ *with* $F_i \subseteq F_{i+1}$. *Given a path* P *in* (T_r, r), *let* $\inf(P)$ *denote the states that are infinitely often visited by* P. *Then* P *is accepted iff the minimal* i *with* $\inf(P) \cap F_i \neq \emptyset$ *is even.*

For two-way alternating automata, the *emptiness problem* is the following question: given a two-way alternating automaton $\mathbf{A}$, is there a tree (T, V) such that $\mathbf{A}$ has an accepting run on (T, V)? It was shown in [40] that this problem is solvable in time that is exponential in the number of $\mathbf{A}$'s states, where the exponent is a polynomial in the length of the parity condition.

3 Hybrid μ-calculus has a tree model property

As usual, when proving a tree model property for the hybrid μ-calculus, we want to "unravel" a given model to a tree model. In the presence of nominals, this is clearly not possible since, for example, the formula $n \wedge \langle\alpha\rangle(m \wedge \langle\beta\rangle n)$ with $n, m \in \mathsf{Nom}$ has no model in the form of a tree. However, we will show that we can unravel each model to a forest, i.e., a collection of trees. When unravelling, we must choose "good" points that witness diamond formulae (i.e., a point y with $y \in \varphi^K$ and $(x, y) \in R(\alpha)$ for $x \in (\langle\alpha\rangle\,\varphi)^K$)—where being "good" is rather tricky in the presence of fixpoints. To this purpose, we define a *choice function* that chooses the "good" witnesses. Essentially, this choice function is a memoryless strategy whose existence is guaranteed for parity games [14]. Definition 4 is the extension of the standard ones to nominals, see, e.g., [35, 40].

Definition 4. *The* closure $\mathsf{cl}(\psi)$ *of a sentence* ψ *is the smallest set of sentences that satisfies the following:*

- $\psi \in \mathsf{cl}(\psi)$,
- *if* $\varphi_1 \wedge \varphi_2 \in \mathsf{cl}(\psi)$ *or* $\varphi_1 \vee \varphi_2 \in \mathsf{cl}(\psi)$, *then* $\{\varphi_1, \varphi_2\} \subseteq \mathsf{cl}(\psi)$,
- *if* $\langle\alpha\rangle\,\varphi \in \mathsf{cl}(\psi)$ *or* $[\alpha]\,\varphi \in \mathsf{cl}(\psi)$, *then* $\varphi \in \mathsf{cl}(\psi)$, *and*
- *if* $\lambda x.\varphi(x) \in \mathsf{cl}(\psi)$, *then* $\varphi(\lambda x.\varphi(x)) \in \mathsf{cl}(\psi)$.

An atom $\mathbf{A} \subseteq \mathsf{cl}(\psi)$ *of* ψ *is a set of formulae that satisfies the following:*

- *if* $p \in \mathsf{AP} \cup \mathsf{Nom}$ *occurs in* ψ, *then, exclusively, either* $p \in \mathbf{A}$ *or* $\neg p \in \mathbf{A}$,
- *if* $\varphi_1 \wedge \varphi_2 \in \mathsf{cl}(\psi)$, *then* $\varphi_1 \wedge \varphi_2 \in \mathbf{A}$ *iff* $\{\varphi_1, \varphi_2\} \subseteq \mathbf{A}$,
- *if* $\varphi_1 \vee \varphi_2 \in \mathsf{cl}(\psi)$, *then* $\varphi_1 \vee \varphi_2 \in \mathbf{A}$ *iff* $\{\varphi_1, \varphi_2\} \cap \mathbf{A} \neq \emptyset$, *and*
- *if* $\lambda x.\varphi(x) \in \mathsf{cl}(\psi)$, *then* $\lambda x.\varphi(x) \in \mathbf{A}$ *iff* $\varphi(\lambda x.\varphi(x)) \in \mathbf{A}$.

The set of atoms of ψ *is denoted* $\mathsf{at}(\psi)$.

A pre-model (K, π) *for a sentence* ψ *consists of a Kripke structure* $K = (W, R, L)$ *and a mapping* $\pi : W \longrightarrow \mathsf{at}(\psi)$ *that satisfies the following properties:*

- *there is a* $u_0 \in W$ *with* $\psi \in \pi(u_0)$,
- *for* $p \in \mathsf{AP} \cup \mathsf{Nom}$, *if* $p \in \pi(u)$, *then* $u \in L(p)$, *and if* $\neg p \in \pi(u)$, *then* $u \notin L(p)$,[2]

[2] Hence if a nominal n is in $\pi(u)$, then $L(n) = \{u\}$.

– *if* $\langle\alpha\rangle\,\varphi \in \pi(u)$*, then there is a* $v \in W$ *with* $(u,v) \in R(\alpha)$ *and* $\varphi \in \pi(v)$*, and*
– *if* $[\alpha]\,\varphi \in \pi(u)$*, then* $\varphi \in \pi(v)$ *for each* $v \in W$ *with* $(u,v) \in R(\alpha)$*.*

A choice function $\mathsf{ch} : W \times \mathsf{cl}(\psi) \longrightarrow \mathsf{cl}(\psi) \cup W$ *for a pre-model* (K,π) *of* ψ *is a partial function that, for each* $u \in W$*,*

(i) if $\varphi_1 \vee \varphi_2 \in \pi(u)$*, then* $\mathsf{ch}(u, \varphi_1 \vee \varphi_2) \in \{\varphi_1, \varphi_2\} \cap \pi(u)$ *and*
(ii) if $\langle\alpha\rangle\,\varphi \in \pi(u)$*, then* $\mathsf{ch}(u, \langle\alpha\rangle\,\varphi) = v$ *for some* v *with* $(u,v) \in R(\alpha)$ *and* $\varphi \in \pi(v)$*.*

An adorned *pre-model* (K,π,ch) *consists of a pre-model* (K,π) *and a choice function* ch*.*

For an adorned pre-model (W,R,L,π,ch) *of* ψ*, the* derivation relation $\rightsquigarrow \subseteq (\mathsf{cl}(\psi), W)^2$ *is defined as follows:*

– *if* $\varphi_1 \vee \varphi_2 \in \pi(u)$*, then* $(\varphi_1 \vee \varphi_2, u) \rightsquigarrow (\mathsf{ch}(\varphi_1 \vee \varphi_2), u)$
– *if* $\varphi_1 \wedge \varphi_2 \in \pi(u)$*, then* $(\varphi_1 \wedge \varphi_2, u) \rightsquigarrow (\varphi_i, u)$ *for each* $i \in \{1,2\}$*,*
– *if* $\langle\alpha\rangle\,\varphi \in \pi(u)$*, then* $(\langle\alpha\rangle\,\varphi, u) \rightsquigarrow (\varphi, \mathsf{ch}(\langle\alpha\rangle\,\varphi, u))$
– *if* $[\alpha]\,\varphi \in \pi(u)$*, then* $([\alpha]\,\varphi, u) \rightsquigarrow (\varphi, v)$ *for each* v *with* $(u,v) \in R(\alpha)$ *(for* $\alpha = o$*, that means that* $([o]\,\varphi, u) \rightsquigarrow (\varphi, v)$ *for each* $v \in W$*)*
– *if* $\lambda x.\varphi(x) \in \pi(u)$*, then* $(\lambda x.\varphi(x), u) \rightsquigarrow (\varphi(\lambda x.\varphi(x)), u)$

A least-fixpoint sentence $\mu x.\varphi(x)$ *is said to be* regenerated *from point* u *to point* v *in an adorned pre-model* (K,π,ch) *if there is a sequence* $(\rho_1,u_1),\ldots,(\rho_k,u_k)$ *with* $k \geq 1$ *such that* $\rho_1 = \rho_k = \mu x.\varphi(x)$*,* $u = u_1$ *and* $v = u_k$*, for each* $1 \leq i < k$*, we have* $(\rho_i,u_i) \rightsquigarrow (\rho_{i+1},u_{i+1})$*, and* $\mu x.\varphi(x)$ *is a sub-sentence of each* ρ_i*. We say that* (K,π,ch) *is* well-founded *if there is no least fixpoint sentence* $\mu x.\varphi(x) \in \mathsf{cl}(\psi)$ *and an infinite sequence* $u_0,u_1,\ldots$ *such that, for each* $i \geq 0$*,* $\mu x.\varphi(x)$ *is regenerated from* u_i *to* u_{i+1}*.*

Lemma 1. *A sentence* ψ *has a model* K *iff* ψ *has a well-founded adorned pre-model* (K,π,ch)*.*

Proof. The construction of a model from a well-founded adorned pre-model and, vice versa, of a well-founded adorned pre-model from a model, are analogous to the constructions that can be found in [35]. These constructions are, as mentioned in [40], insensitive to converse programs, and—due to the according modifications of the technical details—also insensitive to nominals. Indeed, nominals behave simply like atomic propositions provided that $L(n)$ is guaranteed to be interpreted as a singleton. □

Definition 5. *The* relaxation *of a pre-model* (W,R,L,π) *of a sentence* $\psi(n_1, \ldots, n_\ell)$ *consists of mappings* R^r *and* π^r*, where*

$$
\begin{array}{lll}
R^r : \mathsf{Prog} \to W \times W & \text{and} & \\
R^r : \alpha & \mapsto R(\alpha) \setminus \{(u,v) \mid \text{for some } 1 \leq i \leq \ell, L(n_i) = \{v\}\} & \\
\pi^r : W & \to \{G \mid G = G_1 \cup G_2, \quad G_1 \in \mathsf{at}(\psi), \text{ and} & \\
 & \qquad G_2 \subseteq \{\stackrel{\alpha}{\to} n_i \mid \alpha \text{ occurs in } \psi, \alpha \neq o, \text{ and } 1 \leq i \leq \ell\}\} & \\
\pi^r : u & \mapsto \pi(u) \cup \{\stackrel{\alpha}{\to} n \mid (u,v) \in R(\alpha),\ \alpha \neq o, \text{ and } L(n) = \{v\}\} &
\end{array}
$$

A relaxation is a forest *if* R^r *forms a forest.*

Lemma 2. *If a sentence ψ is satisfiable, then it has a well-founded adorned pre-model whose relaxation is a forest and has ψ in the label of one of its roots.*

Proof. Let ψ be satisfiable. Hence there is a well-founded adorned pre-model (K, π, ch) with $K = (W, R, L)$ for ψ due to Lemma 1. Using a technique similar to the one in [40], we construct from (K, π, ch) a well-founded adorned pre-model (K', π', ch') whose relaxation is a forest. Please note that, due to the presence of converse programs, we cannot simply unravel K. However, we can use the choice function to do something similar that yields the desired result also in the presence of converse programs.

Let $\psi = \psi(n_1, \ldots, n_\ell)$ and $w_0 \in W$ such that $w_0 \in \psi^K$. Let $|\psi| = n$, let $\langle \alpha_1 \rangle \varphi_1, \ldots, \langle \alpha_k \rangle \varphi_{k'}$ be all diamond formulae in $\mathsf{cl}(\psi)$, and let k be the maximum of k' and $\ell + 1$. Hence we have $k \leq n$. We define a mapping $\tau : \{1, \ldots, k\}^+ \longrightarrow W \cup \{\bot\}$ inductively, together with an adorned pre-model (K', π', ch') where $K' = (W', R', L')$, $W' = \mathsf{dom}(\tau) \setminus \{x \mid \tau(x) = \bot\}$, and

- for $p \in \mathsf{AP} \cup \mathsf{Nom}$, $x \in L'(p)$ iff $\tau(x) \in L(p)$,
- $\pi'(x) = \pi(\tau(x))$,
- $\mathsf{ch}'(x, \varphi_1 \vee \varphi_2) = \mathsf{ch}(\tau(x), \varphi_1 \vee \varphi_2)$, and
- R' and $\mathsf{ch}'(x, \varphi)$ for diamond formulae φ are defined inductively together with τ.

(Fix the first level) For j with $1 \leq j \leq \ell$, let $v_{f(1)}, \ldots, v_{f(\ell)} \in W$ be such that $L(n_j) = \{v_{f(j)}\}$ and $f(1) \leq \cdots \leq f(\ell) \leq \ell$—since it is possible that $L(n) = L(n')$ for nominals $n \neq n'$, f need not be injective. For $1 \leq j \leq \ell$, set $\tau(f(j)) = v_{f(j)}$.
For $w_0 \in W$ with $w_0 \in \psi^K$, if $w_0 \notin \{v_{f(1)}, \ldots, v_{f(\ell)}\}$, then set $\tau(f(\ell) + 1) = w_0$. Set $\tau(j) = \bot$ for each $1 \leq j \leq k$ not yet defined.

(Fix the rest) For the induction, let i be such that $\tau(x)$ is already defined for each $x \in \{1, \ldots, k\}^i$, and j with $1 \leq j \leq k$ such that $\tau(x1), \ldots, \tau(x(j-1))$ is already defined for each $x \in \{1, \ldots, k\}^i$. Then, for each $x \in \{1, \ldots, k\}^i$, do the following:
(1) if $\langle \alpha_j \rangle \varphi_j \notin \pi'(x)$ or $\tau(x) = \bot$, then define $\tau(xj) = \bot$.
(2) if $\langle \alpha_j \rangle \varphi_j \in \pi'(x)$, then (since (K, π, ch) is a pre-model and $\pi'(x) = \pi(\tau(x))$), there is some $v \in W$ with $\mathsf{ch}(\tau(x), \langle \alpha_j \rangle \varphi_j) = v$ and $(\tau(x), v) \in R(\alpha_j)$.
  - If $\{v\} = L(n_{\ell'})$ for some $1 \leq \ell' \leq \ell$, then (since we have already fixed the first level) there is some r with $1 \leq r \leq \ell$ with $\tau(r) = v$. Add (x, r) to $R'(\alpha_j)$, and set $\mathsf{ch}'(x, \langle \alpha_j \rangle \varphi_j) = r$ and $\tau(xj) = \bot$.
  - Otherwise, add (x, xj) to $R'(\alpha_j)$, set $\tau(xj) = v$ and $\mathsf{ch}'(x, \langle \alpha_j \rangle \varphi_j) = xj$.

Since we started from an adorned pre-model, (K', π', ch') is obviously an adorned pre-model. Moreover, if a sentence $\mu x.\varphi(x)$ is regenerated from x to y in (K', π', ch'), then $\mu x.\varphi(x)$ is also regenerated from $\tau(x)$ to $\tau(y)$ in (K, π, ch). Since the latter is well-founded, we thus have that (K', π', ch') is well-founded. Next, its relaxation R'^r is a forest (consisting of trees starting at the first level) since the only edges in R' that "go back", i.e., that are not of the form (x, xi), are exactly

those that are eliminated in R'^r. Finally, ψ is satisfied in one of the root nodes since, by definition of (K', π', ch'), we have $j \in \psi^{K'}$ for some $1 \leq j \leq f(\ell) + 1$. □

Remark 2. Please note that in this construction, if x satisfies a diamond formula $\langle\alpha\rangle\,\varphi$, then either a successor xj of x or one of the first level nodes representing nominals satisfies φ.

4 Deciding existence of forest models

It remains to devise a procedure that decides, for a sentence ψ, whether it has a well-founded adorned pre-model whose relaxation is a forest. To this purpose, we define a two-way alternating tree automaton that accepts exactly the forest-relaxations of ψ's pre-models—provided that we added a new dummy node whose successors are the root nodes of the forest relaxation.

The automaton depends on a *guess* which contains relevant information concerning the interpretation of nominals. The guess makes sure that the following kind of situation is handled correctly: suppose a nominal n must satisfy a formula of the form $[\alpha]\,\varphi$, and we have a point x with $(x, n) \in R(\overline{\alpha})$, but this relationship is only implicit since we work on relaxations of pre-models, i.e., $(x, n) \notin R^r(\overline{\alpha})$ and $\overset{\overline{\alpha}}{\to} n \in \pi^r(x)$. In that case, the guess makes sure that x satisfies φ since it determines which box formulae are satisfied by nominals. Moreover, the guess determines which nominals are interpreted as the same objects, and how nominals are related to each other by programs.

It is possible to refer all this "guessing" directly to the automaton—hence we had only one automaton instead of one per guess. We have chosen, however, to work with explicit guesses since, on the one hand, it makes explicit the additional non-determinism one has to cope with in the presence of nominals and how it can be dealt with. On the other hand and more importantly, referring the guessing into the automaton would yield a quadratic blow-up of the state space. Let n be the number of states and m be the length of the acceptance condition of a two-way alternating tree automaton. When deciding emptiness of a two-way alternating tree automaton [40], it is transformed into a non-deterministic (one-way) parity tree automaton whose state space is of size $(nm^2)^{nm^2}$, and whose acceptance condition is of length nm^2. Emptiness of the latter automaton can be decided in time $2^{\mathcal{O}((n^2m^4)(\log n + 2\log m))}$ [25]. Hence a (quadratic) blow-up of the state space of our initial two-way alternating tree automaton would further increase the degree of the polynomial in the exponent of the runtime, and thus be rather expensive.

Formally, a guess consists of three components, the first one consisting, for each nominal n, of a set γ of formulae satisfied by a point u with $L(n) = \{u\}$. Since one point may represent several nominals, we use a second component f to relate a nominal n_i to "its" set of formulae $\gamma_{f(i)}$. The third component describes how two points representing nominals are interrelated via (interpretations of) programs, making sure that, if one is an α-successor of the other, then the other is an $\overline{\alpha}$-successor of the first one.

Definition 6. *A* guess $\mathcal{G} = (G, f, C)$ *for a hybrid* μ*-calculus sentence* $\psi(n_1, \ldots, n_\ell)$ *consists of a* guess list $G = (\gamma_1, \ldots, \gamma_\ell)$ *together with* connections $C \subseteq \mathsf{Nom} \times \mathsf{Prog}_\psi \times \mathsf{Nom}$ *and a* guess mapping $f : \{1, \ldots, \ell\} \longrightarrow \{1, \ldots, \ell\}$*, where, for each* $1 \leq i, j \leq \ell$*, we have* $\emptyset \subsetneq \gamma_i \subseteq \mathsf{cl}(\psi)$ *or* $\gamma_i = \bot$*,* $n_i \in \gamma_{f(i)}$*,* $n_i \notin \gamma_j$ *for all* $j \neq f(i)$*,* $\mathsf{Nom} \cap \gamma_i = \emptyset$ *implies* $\gamma_i = \bot$*, and* $(n_i, \alpha, n_j) \in C$ *iff* $(n_j, \overline{\alpha}, n_i) \in C$*.*

Theorem 1. *Let* ψ *be a hybrid* μ*-calculus sentence. For each guess* $\mathcal{G}$ *for* ψ*, we define a two-way alternating tree automaton* $\mathcal{B}(\psi, \mathcal{G})$*, such that*

1. *if* ψ *is satisfiable, then there exists a guess* $\mathcal{G}'$ *for* ψ *such that the language accepted by* $\mathcal{B}(\psi, \mathcal{G}')$ *is non-empty,*
2. *if a tree is accepted by* $\mathcal{B}(\psi, \mathcal{G})$*, then eliminating its root node yields a forest relaxation of a well-founded adorned pre-model of* ψ*, and*
3. *the number of* $\mathcal{B}(\psi, \mathcal{G})$*'s states is linear in* $|\psi|$*.*

Proof. For ease of presentation, we assume that all input trees are full trees, i.e., all non-leaf nodes have the same number of children. As we have seen in the proof of Lemma 2, we can simply "fill" a tree with additional nodes labelled $\bot$ to make it a full tree. Moreover, we assume a "dummy" root node whose direct successors are exactly the root nodes of trees in the forest relaxation.

For a sentence $\psi(n_1, \ldots, n_\ell)$ with k' diamond subformulae in $\mathsf{cl}(\psi)$ as specified in the proof of Lemma 2 and a guess $\mathcal{G}$, we define two alternating automata, $\mathcal{A}(\psi, \mathcal{G})$ and $\tilde{\mathcal{A}}(\psi, \mathcal{G})$, and then define $\mathcal{B}(\psi, \mathcal{G})$ as the intersection of $\mathcal{A}(\psi, \mathcal{G})$ and $\tilde{\mathcal{A}}(\psi, \mathcal{G})$. For alternating automata, intersection is trivial (basically, we introduce a new initial state $\tilde{q}$ with $\delta(\tilde{q}, \sigma) = (0, q_0) \wedge (0, q_0')$ for the former initial states q_0, q_0'), and the size of $\mathcal{B}(\psi, \mathcal{G})$ is the sum of the sizes of $\mathcal{A}(\psi, \mathcal{G})$ and $\tilde{\mathcal{A}}(\psi, \mathcal{G})$.

The automaton $\tilde{\mathcal{A}}(\psi, \mathcal{G})$ is rather simple and guarantees that the structure of the input tree is as required, whereas $\mathcal{A}(\psi, \mathcal{G})$ really makes sure that the input tree (more precisely, the sub-forest of the input tree obtained by eliminating the root and all nodes labelled with $\bot$) is a relaxation of a well-founded adorned pre-model.

Both automata work on the same alphabet Σ, which is defined as follows: For $\mathsf{Prog}^+ = \{p_\alpha, p_{\overline{\alpha}}, \overline{p}_\alpha, \overline{p}_{\overline{\alpha}} \mid \alpha \text{ is a program in } \psi \text{ different from } o\}$,

$$\Sigma = \{\bot, \mathsf{root}\} \cup \{\sigma \mid \begin{array}{l} \sigma \subseteq \mathsf{AP} \cup \mathsf{Nom} \cup \mathsf{Prog}^+ \cup \{\stackrel{\alpha_j}{\rightarrow} n_i \mid 1 \leq j \leq m \text{ and } 1 \leq i \leq \ell\}, \\ \quad \sigma \text{ contains, for each } \alpha, \text{ exclusively, either } p_\alpha \text{ or } \overline{p}_\alpha, \text{ and,} \\ \qquad \text{exclusively, either } p_{\overline{\alpha}} \text{ or } \overline{p}_{\overline{\alpha}}\} \end{array}$$

The intuition of the additional symbols are as follows: Nodes not representing points in a Kripke structure are labelled root and $\bot$, where root labels the root node. Nodes having n_i (i.e., the node labelled with the corresponding guess $\gamma_{f(i)}$) as an α-successor are marked $\stackrel{\alpha}{\rightarrow} n_i$, just like in relaxations. A node label contains p_α ($p_{\overline{\alpha}}$) if this node is an α-successor ($\overline{\alpha}$-successor) of its (unique) predecessor. We do allow that a node is both an α- and a β-successor, or that no program can be associated to the edge between two nodes. Analogously, $\overline{p}_\alpha$ ($\overline{p}_{\overline{\alpha}}$) are used to mark those nodes that are *not* α-successors ($\overline{\alpha}$-successors).

The "simple" automaton $\tilde{\mathcal{A}}(\psi, \mathcal{G})$ guarantees that root is only found at the root label, the nominals in γ_i are only found at the ith successors of the root, the

first level nodes contain no p_α or $p_{\overline{\alpha}}$ and that, if a nominal n_i has another nominal n_j as its α-successor (i.e., if $\stackrel{\alpha}{\rightarrow} n_j$ is in the label of the node representing n_i), then n_j has n_i as its $\overline{\alpha}$-successor (i.e., $\stackrel{\alpha}{\rightarrow} n_i$ is in the label of the node representing n_j). More precisely, $\tilde{\mathcal{A}}(\psi, \mathcal{G}) = (\Sigma, \{q_0, q_1, \ldots, q_\ell, q', q\}, \delta', q_0)$ is a safety one-way alternating automaton (i.e., each state is accepting and thus every run is an accepting run), and δ' is defined as follows for $\sigma \in \Sigma$:

$$\delta'(q_0, \sigma) = \begin{cases} \bigwedge_{i=1}^{\ell}(i, q_i) \wedge \bigwedge_{i=\ell+1}^{k}(i, q) \wedge \bigwedge_{i=1}^{k}(i, q') & \text{if } \mathsf{root} = \sigma \\ \mathbf{false} & \text{otherwise} \end{cases}$$

$$\delta'(q', \sigma) = \begin{cases} \mathbf{true} & \text{if } p_\alpha \notin \sigma \text{ and } p_{\overline{\alpha}} \notin \sigma \text{ for each } \alpha \neq o \text{ in } \psi \\ \mathbf{false} & \text{otherwise} \end{cases}$$

for $1 \leq i \leq \ell$:

$$\delta'(q_i, \sigma) = \begin{cases} \bigwedge_{i=1}^{k}(i, q) & \text{if } \gamma_i \cap (\mathsf{Nom} \cup \mathsf{AP}) = \sigma \cap (\mathsf{Nom} \cup \mathsf{AP}),\ \mathsf{root} \neq \sigma, \text{ and,} \\ & \text{for each } n \in \mathsf{Nom} \cap \sigma \text{ and } (n, \alpha, n') \in C,\ \stackrel{\alpha}{\rightarrow} n' \in \sigma \\ \mathbf{false} & \text{otherwise} \end{cases}$$

$$\delta'(q, \sigma) = \begin{cases} \bigwedge_{i=1}^{k}(i, q) & \text{if } \sigma \cap \mathsf{Nom} = \emptyset \text{ and } \mathsf{root} \neq \sigma \\ \mathbf{false} & \text{otherwise} \end{cases}$$

Due to the symmetry in the definition of the connection component in a guess and the way $\delta'(q_i, \sigma)$ is defined, if $\tilde{\mathcal{A}}(\psi, \mathcal{G})$ accepts a tree, $\{n_i, \stackrel{\alpha}{\rightarrow} n_j\} \subseteq \sigma$, and $n_j \in \sigma'$, then $\stackrel{\overline{\alpha}}{\rightarrow} n_i \in \sigma'$, and σ, σ' label direct successors of the root node.

The two-way alternating tree automaton $\mathcal{A}(\psi, \mathcal{G})$ verifies that the input tree is indeed a relaxation of a well-founded adorned pre-model. To this purpose, (most of) its states correspond to formulae in $\mathsf{cl}(\psi)$, and the transition relation basically follows the semantics.

The first conjunct in the definition of $\delta(q'_0, \sigma)$ guarantees that the ith successor of the root node indeed satisfies all formulae in γ_i, and that one of the root node successors satisfies ψ.

An additional state q' that "travels" once through the whole input tree makes sure that, whenever a node has a nominal n_i as its implicit $\overline{\alpha}$-successor (i.e., its label contains $\stackrel{\overline{\alpha}}{\rightarrow} n_i$), then this node satisfies indeed all formulae φ with $[\alpha]\,\varphi \in \gamma_{f(i)}$.

Finally, the diamond and box formulae on the universal role are treated separately since they apply to all but the root node, regardless of marks p_α or $\overline{p}_\alpha$. Please note that, since the root node does not represent any point of a Kripke structure, $\delta([o]\,\varphi, \mathsf{root})$ is defined such that only all root successors satisfy $[o]\,\varphi$, but not the root node itself. More precisely, we have

$$\begin{aligned} \mathcal{A}(\psi, \mathcal{G}) &= (\Sigma, Q, \delta, q'_0, F), \text{ with} \\ Q &= \{\bot, q'_0, q'\} \cup \mathsf{cl}(\psi) \cup \mathsf{Prog}^+. \end{aligned}$$

The transition relation δ is defined as follows: firstly, for $q \in Q$ and $\sigma \in \Sigma$ let

$$\delta(q, \bot) = \begin{cases} \mathbf{true} & \text{if } q = \bot \\ \mathbf{false} & \text{otherwise} \end{cases} \qquad \delta(\bot, \sigma) = \begin{cases} \mathbf{true} & \text{if } \sigma = \bot \\ \mathbf{false} & \text{otherwise} \end{cases}$$

Secondly, for $1 \leq i \leq \ell$ and $\sigma \in \Sigma$, let

$$\Gamma(i) = \begin{cases} (i, \bot) & \text{if } \gamma_i = \bot \\ \bigwedge_{\varphi \in \gamma_i} (i, \varphi) & \text{if } \gamma_i \subseteq \mathsf{cl}(\psi) \end{cases} \qquad N(\sigma) = \bigwedge_{\stackrel{\overline{\alpha}}{\to} n_i \in \sigma \text{ and } [\alpha]\varphi \in \gamma_{f(i)}} (0, \varphi)$$

Thirdly, for $\sigma \in \Sigma$, $\sigma \neq \bot$, and α a program, we define δ as follows:

$$\delta(q_0', \sigma) = \textstyle\bigwedge_{i=1}^{\ell} \Gamma(i) \wedge \bigvee_{i=1}^{k} (i, \psi) \wedge \bigwedge_{i=1}^{k} ((i, q') \vee (i, \bot))$$
$$\delta(q', \sigma) = N(\sigma) \wedge \textstyle\bigwedge_{i=1}^{k} ((i, q') \vee (i, \bot))$$

for $p \in \mathsf{AP} \cup \mathsf{Nom} \cup \mathsf{Prog}^+$:

$$\delta(p, \sigma) = \begin{cases} \mathbf{true} \text{ if } p \in \sigma \\ \mathbf{false} \text{ otherwise} \end{cases}$$

for $p \in \mathsf{AP} \cup \mathsf{Nom}$:

$$\delta(\neg p, \sigma) = \begin{cases} \mathbf{true} \text{ if } p \notin \sigma \text{ and } \sigma \neq \mathsf{root} \\ \mathbf{false} \text{ otherwise} \end{cases}$$

$$\delta(\varphi_1 \wedge \varphi_2, \sigma) = (0, \varphi_1) \wedge (0, \varphi_2)$$
$$\delta(\varphi_1 \vee \varphi_2, \sigma) = (0, \varphi_1) \vee (0, \varphi_2)$$
$$\delta(\lambda x.\varphi(x), \sigma) = (0, \varphi(\lambda x.\varphi(x)))$$

for $\alpha \notin \{o, o^-\}$:

$$\delta(\langle\alpha\rangle \varphi, \sigma) = \begin{cases} \mathbf{true} & \text{if } \stackrel{\alpha}{\to} n_i \in \sigma \text{ and } \varphi \in \gamma_{f(i)} \\ \bigvee_{j=1}^{k} ((j, \varphi) \wedge (j, p_\alpha)) & \text{otherwise} \end{cases}$$

for $\alpha \notin \{o, o^-\}$:

$$\delta([\alpha] \varphi, \sigma) = \begin{cases} \mathbf{false} & \text{if } \stackrel{\alpha}{\to} n_i \in \sigma \text{ and } \varphi \notin \gamma_{f(i)} \\ ((-1, \varphi) \vee (0, \overline{p}_{\overline{\alpha}})) \wedge & \text{otherwise} \\ \bigwedge_{j=1}^{k} ((j, \varphi) \vee (j, \overline{p}_\alpha) \vee (j, \bot)) & \end{cases}$$

for $\alpha \in \{o, o^-\}$:

$$\delta(\langle\alpha\rangle \varphi, \sigma) = \begin{cases} \mathbf{true} & \text{if } \varphi \in \gamma_{f(i)} \\ \bigvee_{j=1}^{k} (j, \varphi) & \text{otherwise} \end{cases}$$

for $\alpha \in \{o, o^-\}$:

$$\delta([\alpha] \varphi, \sigma) = \begin{cases} (0, \varphi) \wedge (-1, [\alpha]\varphi) \wedge & \text{if } \mathsf{root} \neq \sigma \\ \bigwedge_{j=1}^{k} ((j, [\alpha]\varphi) \vee (j, \bot)) & \\ \bigwedge_{j=1}^{k} ((j, [\alpha]\varphi) \vee (j, \bot)) & \text{otherwise} \end{cases}$$

Please note that, following the construction in the proof of Lemma 2, satisfaction of diamond formulae (including those on the universal program) needs to be tested for only in direct successors and in the nodes representing nominals. Moreover, since $\psi = \psi(n_1, \ldots, n_\ell)$ and due to the definition of $\delta(q_0', \sigma)$ and $\Gamma(i)$, δ checks whether the node representing n_i satisfies indeed all formulae in $\gamma_{f(i)}$.

The acceptance condition F is defined analogously to the one in [15, 24], and given here for the sake of completeness. Firstly, for a fixpoint formula $\varphi \in \mathsf{cl}(\psi)$, define the alternation level of φ to be the number of alternating fixpoint formulae one has to "wrap φ with" to reach a sub-*sentence* of ψ. More precisely, the

alternation level $\mathsf{al}_\psi(\varphi)$ of $\varphi = \lambda x.\varphi'(x) \in \mathsf{cl}(\psi)$ is defined as follows [3]: if φ is a sentence, then $\mathsf{al}_\psi(\varphi) = 1$. Otherwise, let $\rho = \lambda' y.\rho'(y)$ be the innermost fixpoint formula in $\mathsf{cl}(\psi)$ that contains φ as a proper sub-formula. If $\lambda = \lambda'$, then $\mathsf{al}_\psi(\varphi) = \mathsf{al}_\psi(\rho)$, otherwise $\mathsf{al}_\psi(\varphi) = \mathsf{al}_\psi(\rho) + 1$. Let d be the maximal alternation level of (fixpoint) subformulae of ψ, and define

$$G_i = \{\nu x.\varphi(x) \in \mathsf{cl}(\psi) \mid \mathsf{al}_\psi(\nu x.\varphi(x)) = i\}$$
$$L_i = \{\mu x.\varphi(x) \in \mathsf{cl}(\psi) \mid \mathsf{al}_\psi(\mu x.\varphi(x)) \leq i\}$$

Now we are ready to define the acceptance condition $F = \{F_1, \ldots, F_{2d}\}$ with $F_i = \emptyset$ for $i = 0$, $F_i = F_{i-1} \cup L_i$ for odd $i \geq 1$, and $F_i = F_{i-1} \cup G_i$ for even $i \geq 1$. Obviously, $F_i \subseteq F_{i+1}$ for each $1 \leq i \leq 2d$. As mentioned in Definition 3, a path r_p of a run r is accepting if the minimal i with $\inf(r_p) \cap F_i \neq \emptyset$ is even—this i corresponds to the outermost fixpoint formula that was infinitely often visited/postponed. A run r is accepting if each of its paths are accepting. Intuitively, the acceptance condition makes sure that, if a fixpoint formula was visited infinitely often, then this was a greatest fixpoint formulae, and that all of its least fixpoint super-formulae were visited only finitely many times.

It remains to verify the three claims in Theorem 1. The proof of the first one uses Lemma 1 and a straightforward construction of a guess $\mathcal{G}$ from a forest relaxation of a well-founded adorned pre-model, and then shows how an input forest similar to the one constructed in the proof of Lemma 1 is accepted by $\mathcal{B}(\psi, \mathcal{G})$. The second claim can be proved by taking an accepting run of $\mathcal{B}(\psi, \mathcal{G})$ on some input tree, and verifying that the input tree indeed satisfies all properties of relaxations of well-founded adorned pre-models. Finally, the third claim is by definition of $\mathcal{B}(\psi, \mathcal{G})$. □

Theorem 2. *Satisfiability of hybrid μ-calculus is decidable in exponential time.*

Proof. As we have mentioned in the beginning of Section 4, emptiness of $\mathcal{B}(\psi, \mathcal{G})$ can be decided in time $2^{O(n^6 \log n)}$ for $n = |\psi|$. Let ℓ be the number of nominals and m the number of programs different from o in ψ. Since, for a guess $\mathcal{G} = (G, f, C)$, the mapping f is determined by G, the number of guesses is bound by the number of connections and guess lists, i.e., by $2^{\ell^2 m} \cdot 2^{\ell n}$. Hence we have to test at most an exponential number of automata $\mathcal{B}(\psi, \mathcal{G})$ for emptiness. Combining these results with Lemma 1, Lemma 2, and Theorem 1 concludes the proof. □

5 Conclusion

We have shown that satisfiability of the hybrid μ-calculus can be decided in exponential time, thus partially answering an open question in [5]. Deciding satisfiability of a logic that lacks the tree model property using tree automata was possible using a certain abstraction of models, *relaxations*, and involved an additional non-determinism, *guesses*. Then, we were able to use the emptiness algorithm in [40] as a sub-routine. For an input sentence, the algorithm presented constructs a family of tree automata, each of which depends on a guess that

determines relevant information concerning the interpretation of nominals. We have chosen this explicit guess since, on the one hand, it directly shows how nominals can be dealt with. On the other hand, when referring the guessing into the automaton, we would blow up its state space quadratically. Since deciding emptiness of this family of automata is exponential in the size of its state space, it is clearly preferable to avoid even such a polynomial blow-up. The complexity of the hybrid μ-calculus with deterministic programs[3] remains an interesting open problem. As a consequence of NExpTime-hardness results in [37], this extension leads to NExpTime-hardness. Another interesting research problem is the development of practical decision procedures for (fragments of) the hybrid μ-calculus. To the best of our knowledge, automata-theoretic methods are the only known methods for the μ-calculus, and, so far, such methods have been implemented successfully only for linear temporal logic, see, e.g., [8, 20].

References

1. C. Areces, P. Blackburn, and M. Marx. The computational complexity of hybrid temporal logics. *Logic Journal of the IGPL*, 8(5), 2000.
2. F. Baader and B. Hollunder. A terminological knowledge representation system with complete inference algorithm. In *Proc. of PDK-91*, vol. 567 of *LNAI*. Springer-Verlag, 1991.
3. G. Bhat and R. Cleaveland. Efficient local model-checking for fragments of the modal μ-calculus. In *Proc. of TACAS*, vol. 1055 of *LNCS*. Springer-Verlag, 1996.
4. P. Blackburn. Nominal tense logic. *Notre Dame Journal of Formal Logic*, 34, 1993.
5. D. Calvanese, G. De Giacomo, and M. Lenzerini. Reasoning in expressive description logics with fixpoints based on automata on infinite trees. In *Proc. of IJCAI'99*, 1999.
6. D. Calvanese, G. De Giacomo, M. Lenzerini, D. Nardi, and R. Rosati. Description logic framework for information integration. In *Proc. of KR-98*, 1998.
7. D. Calvanese, M. Lenzerini, and D. Nardi. Description logics for conceptual data modeling. In *Logics for Databases and Information Systems*. Kluwer Academic Publisher, 1998.
8. E.M. Clarke, O. Grumberg, and K. Hamaguchi. Another look at LTL model checking. In *Proc. of CAV'94*, vol. 818 of *LNCS*, pages 415–427. Springer-Verlag, 1994.
9. G. De Giacomo and M. Lenzerini. Boosting the correspondence between description logics and propositional dynamic logics. In *Proc. of AAAI-94*, 1994.
10. G. De Giacomo and M. Lenzerini. Concept language with number restrictions and fixpoints, and its relationship with μ-calculus. In *Proc. of ECAI-94*, 1994.
11. G. De Giacomo and M. Lenzerini. Tbox and Abox reasoning in expressive description logics. In *Proc. of KR-96*. Morgan Kaufmann, 1996.
12. F. Donini, M. Lenzerini, D. Nardi, and W. Nutt. The complexity of concept languages. In *Proc. of KR-91*. Morgan Kaufmann, 1991.
13. F. M. Donini, M. Lenzerini, D. Nardi, and W. Nutt. The complexity of concept languages. *Information and Computation*, 134, 1997.
14. E. A. Emerson and C. S. Jutla. Tree automata, μ-calculus, and determinacy. In *Proc. of FOCS-91*. IEEE, 1991.

[3] Or Description Logic's number restrictions or Modal Logic's graded modalities.

15. E. A. Emerson, C. S. Jutla, and A. P. Sistla. On model checking for fragments of the μ-calculus. In *Proc. of CAV'93*, vol. 697 of *LNCS*. Springer-Verlag, 1993.
16. D. Fensel, I. Horrocks, F. van Harmelen, S. Decker, M. Erdmann, and M. Klein. OIL in a nutshell. In *Proc. EKAW-2000*, vol. 1937 of LNAI, 2000. Springer-Verlag.
17. K. Fine. In so many possible worlds. *Notre Dame J. of Formal Logics*, 13, 1972.
18. E. Franconi and U. Sattler. A data warehouse conceptual data model for multidimensional aggregation: a preliminary report. *AI*IA Notizie*, 1, 1999.
19. V. Haarslev and R. Möller. Expressive abox reasoning with number restrictions, role hierarchies, and transitively closed roles. In *Proc. of KR-00*, 2000.
20. Gerard J. Holzmann. The spin model checker. *IEEE Trans. on Software Engineering*, 23(5), 1997.
21. I. Horrocks. Using an Expressive Description Logic: FaCT or Fiction? In *Proc. of KR-98*, 1998.
22. I. Horrocks, U. Sattler, and S. Tobies. Practical reasoning for very expressive description logics. *Logic Journal of the IGPL*, 8(3), May 2000.
23. D. Kozen. Results on the propositional μ-calculus. In *Proc. of ICALP'82*, vol. 140 of *LNCS*. Springer-Verlag, 1982.
24. O. Kupferman and M. Y. Vardi. μ-calculus synthesis. In *Proc. MFCS'00*, LNCS. Springer-Verlag, 2000.
25. O. Kupferman and M.Y. Vardi. Weak alternating automata and tree automata emptiness. In *Proc. of STOC-98*, 1998.
26. H. Levesque and R. J. Brachman. Expressiveness and tractability in knowledge representation and reasoning. *Computational Intelligence*, 3, 1987.
27. D. E. Muller and P. E. Schupp. Alternating automata on infinite trees. *Theoretical Computer Science*, 54(1-2), 1987.
28. B. Nebel. *Reasoning and Revision in Hybrid Representation Systems.* LNAI. Springer-Verlag, 1990.
29. P. F. Patel-Schneider and I. Horrocks. DLP and FaCT. In *Proc. TABLEAUX-99*, vol. 1397 of *LNAI*. Springer-Verlag, 1999.
30. A. Prior. *Past, Present and Future.* Oxford University Press, 1967.
31. A. Rector and I. Horrocks. Experience building a large, re-usable medical ontology using a description logic with transitivity and concept inclusions. In *Proc. of the AAAI Spring Symposium on Ontological Engineering.* AAAI Press, 1997.
32. K. Schild. A correspondence theory for terminological logics: Preliminary report. In *Proc. of IJCAI-91*, 1991.
33. K. Schild. Terminological cycles and the propositional μ-calculus. In *Proc. of KR-94*, 1994. Morgan Kaufmann.
34. M. Schmidt-Schauß and G. Smolka. Attributive concept descriptions with complements. *Artificial Intelligence*, 48(1), 1991.
35. R. S. Streett and E. A. Emerson. An automata theoretic decision procedure for the propositional μ-calculus. *Information and Computation*, 81(3), 1989.
36. W. Thomas. Languages, automata, and logic. In *Handbook of Formal Language Theory*, vol 1. Springer-Verlag, 1997.
37. S. Tobies. The complexity of reasoning with cardinality restrictions and nominals in expressive description logics. *J. of Artificial Intelligence Research*, 12, 2000.
38. S. Tobies. PSPACE reasoning for graded modal logics. *J. of Logic and Computation*, 2001. To appear.
39. M. Y. Vardi. What makes modal logic so robustly decidable? In *Descriptive Complexity and Finite Models*, American Mathematical Society, 1997.
40. M. Y. Vardi. Reasoning about the past with two-way automata. In *Proc. of ICALP'98*, vol. 1443 of *LNCS*, 1998. Springer-Verlag.