The Complexity of One-Agent Refinement Modal Logic

Laura Bozzelli[1], Hans van Ditmarsch[2], and Sophie Pinchinat[3]

[1] Technical University of Madrid (UPM), Madrid, Spain
laura.bozzelli@fi.upm.es
[2] Logic, University of Sevilla, Spain & IMSc, Chennai, India
hvd@us.es
[3] IRISA/INRIA, University of Rennes
Sophie.Pinchinat@irisa.fr

Abstract. We investigate the complexity of satisfiability for one-agent Refinement Modal Logic (RML), a known extension of basic modal logic (ML) obtained by adding refinement quantifiers on structures. It is known that RML has the same expressiveness as ML, but the translation of RML into ML is of non-elementary complexity, and RML is at least *doubly* exponentially more succinct than ML. In this paper, we show that RML-satisfiability is 'only' *singly* exponentially harder than ML-satisfiability, the latter being a well-known PSPACE-complete problem. More precisely, we establish that RML-satisfiability is complete for the complexity class $\mathsf{AEXP}_{\mathsf{pol}}$, i.e., the class of problems solvable by alternating Turing machines running in single exponential time but only with a polynomial number of alternations (note that $\mathsf{NEXPTIME} \subseteq \mathsf{AEXP}_{\mathsf{pol}} \subseteq \mathsf{EXPSPACE}$).

1 Introduction

From propositional to refinement quantification in modal logics. Modal logics with propositional quantifiers have been investigated since Fine's seminal paper [7]. Fine distinguishes different propositional quantifications, which allow different kinds of model transformations, not all of which are, in our modern terms, bisimulation preserving. However, in the general case, propositional quantification can easily lead to undecidable logics [7,8]. This has motivated, more recently, the introduction of bisimulation quantified logics [19,10,8,15]: in this framework, the quantification is over the models which are bisimilar to the current model except for a propositional variable p (note that this includes model restriction). A novel way of quantification in rather dynamic modal logics is quantifying over all modally definable submodels of a given model [2]. The setting for these logics is how to quantify over information change; for example, in the logic APAL of [2], an expression that we might write as $\exists_r \varphi$ for our purposes stands for "there is a formula ψ such that after model restriction with relativization to ψ, the formula φ holds". *Refinement modal logic* (see [17,18] and the unpublished manuscript [4]) is a generalization of this perspective to more complex model transformations than mere model restrictions: this is achieved by existential and universal quantifiers which range over the *refinements* of the current structure (model). From the *atoms/forth/back* requirements of bisimulation, a refinement of a modal structure need only satisfy *atoms* and *back*. It is the dual of a simulation that need only satisfy *atoms* and *forth*. Refinement is more general than model restriction, since it is equivalent to bisimulation followed by model restriction.

L. Fariñas del Cerro, A. Herzig, and J. Mengin (Eds.): JELIA 2012, LNCS 7519, pp. 120–133, 2012.

Moreover, refinement quantification corresponds to implicit quantification over propositional variables (i.e., quantification over variables not occurring in the formula bound by the quantifier), just as in bisimulation quantified logics we have explicit quantification over propositional variables; in fact, it is an abstraction of a bisimulation quantification followed by a relativization [4]. As amply illustrated in [4], refinement quantification has applications in many settings: in logics for games [1,15], it may correspond to a player discarding some moves; for program logics [9], it may correspond to operational refinement; and for logics for spatial reasoning, it may correspond to sub-space projections [14].

Our Contribution. We focus on complexity issues for (one-agent) Refinement Modal Logic (RML) [17,18,4], the extension of (one-agent) basic modal logic (ML) obtained by adding the existential and universal refinement quantifiers $\exists_r$ and $\forall_r$. It is known [18,4] that RML has the same expressivity as ML, but the translation of RML into ML is of non-elementary complexity and no elementary upper bound is known for its satisfiability problem [4]. In fact, an upper bound in 2EXPTIME has been claimed in [18] by a tableaux-based procedure: the authors later concluded that the procedure is sound but not complete [4]. In this paper, our aim is to close that gap. We also investigate the complexity of satisfiability for some equi-expressive fragments of RML. In particular, we associate with each RML formula φ a parameter $\Upsilon_w(\varphi)$ corresponding to a slight variant of the classical quantifier alternation depth (measured w.r.t. $\exists_r$ and $\forall_r$), and for each $k \geq 1$, we consider the fragment RML^k consisting of the RML formulas φ such that $\Upsilon_w(\varphi) \leq k$. Moreover, we consider the existential (resp., universal) fragment $\mathsf{RML}^{\exists}$ (resp., $\mathsf{RML}^{\forall}$) obtained by disallowing the universal (resp., existential) refinement quantifier.

In order to present our results, first, we recall some computational complexity classes. We assume familiarity with the standard notions of complexity theory [11,13]. We will make use of the levels Σ_k^{EXP} ($k \geq 1$) of the exponential-time hierarchy EH, which are defined similarly to the levels Σ_k^{P} of the polynomial-time hierarchy PH, but with NP replaced with NEXPTIME. In particular, Σ_k^{EXP} corresponds to the class of problems decided by single exponential-time bounded Alternating Turing Machines (ATM, for short) with at most $k-1$ alternations and where the initial state is existential [11]. Note that $\Sigma_1^{\mathsf{EXP}} = \mathsf{NEXPTIME}$. Recall that $\mathsf{EH} \subseteq \mathsf{EXPSPACE}$ and EXPSPACE corresponds to the class of problems decided by single exponential-time bounded ATM (with no constraint on the number of alternations) [5]. We are also interested in an intermediate class between EH and EXPSPACE, here denoted by $\mathsf{AEXP}_{\mathsf{pol}}$, that captures the precise complexity of some relevant problems [6,11,16] such as the first-order theory of real addition with order [6,11]. Formally, $\mathsf{AEXP}_{\mathsf{pol}}$ is the class of problems solvable by single exponential-time bounded ATM with a polynomial-bounded number of alternations.

Our complexity results are summarized in Figure 1 where we also recall the well-known complexity of ML-satisfiability. For the upper bounds, the (technically non-trivial) main step in the proposed approach exploits a "small" size model property: we establish that like basic modal logic ML, RML enjoys a single exponential size model property.[1]

We conclude this section by observing that our results are surprising for the following reason. While our results essentially indicate that satisfiability of RML is "only" *singly* exponentially harder than satisfiability of ML, it is known [4] that RML is *doubly* exponentially more succinct than ML.

[1] Omitted proofs can be found in the online extended version with the same title.

ML	$\mathsf{RML}^{\exists} = \mathsf{RML}^{1}$	$\mathsf{RML}^{\forall} \subseteq \mathsf{RML}^{2}$	RML^{k+1} $(k \geq 1)$	RML
PSPACE-complete	$\in$ NEXPTIME PSPACE-hard	$\in \Sigma_2^{\mathsf{EXP}}$ NEXPTIME-hard	$\in \Sigma_{k+1}^{\mathsf{EXP}}$ Σ_k^{EXP}-hard	$\mathsf{AEXP}_{\mathsf{pol}}$-complete

Fig. 1. Complexity results for satisfiability of RML and RML-fragments

2 Preliminaries

In the rest of this section, we fix a finite set P of atomic propositions.

Structures, Tree Structures, and Refinement Preorder. A (*one-agent Kripke*) *structure* (*over* P) is a tuple $M = \langle S, E, V \rangle$, where S is a set of states (or worlds), $E \subseteq S \times S$ is a transition (or accessibility) relation, and $V : S \mapsto 2^P$ is a *P-valuation* assigning to each state s the set of propositions in P which hold at s. For states s and t of M such that $(s,t) \in E$, we say that t is a *successor* of s. A *pointed* structure is a pair (M,s) consisting of a structure M and a designated initial state s of M.

A tree T is a prefix-closed subset of $\mathbb{N}^*$, where $\mathbb{N}$ is the set of natural numbers. The elements of T are called *nodes* and the empty word ε is the *root* of T. For $x \in T$, the set of children (or successors) of x is $\{x \cdot i \in T \mid i \in \mathbb{N}\}$. The *size* $|T|$ of T is the number of T-nodes. A (*rooted*) *tree structure* (over P) is a pair $\langle T, V \rangle$ such that T is a tree and $V : T \mapsto 2^P$ is a P-valuation over T. For $x \in T$, the *tree substructure of* $\langle T, V \rangle$ *rooted at* x is the tree structure $\langle T_x, V_x \rangle$, also denoted by $\langle T, V \rangle_x$, where $T_x = \{y \in \mathbb{N}^* \mid x \cdot y \in T\}$ and $V_x(y) = V(x \cdot y)$ for all $y \in T_x$. Note that a tree structure $\langle T, V \rangle$ corresponds to the pointed structure $(\langle T, E, V \rangle, \varepsilon)$, where $(x,y) \in E$ iff y is a child of x. Moreover, we can associate with any pointed structure (M,s) a tree structure, denoted by $Unw(M,s)$, obtained by unwinding M from s in the usual way.

For two structures $M = \langle S, E, V \rangle$ and $M' = \langle S', E', V' \rangle$, a *refinement from* M *to* M' is a relation $\mathfrak{R} \subseteq S \times S$ such that for all $(s,s') \in \mathfrak{R}$: (i) $V(s) = V'(s')$, and (ii) if $(s',t') \in E'$ for some $t' \in S'$, then there is some state $t \in S$ such that $(s,t) \in E$ and $(t,t') \in \mathfrak{R}$. If, additionally, the inverse of $\mathfrak{R}$ is a refinement from M' to M, then $\mathfrak{R}$ is a *bisimulation* from M to M'. For states $s \in S$ and $s' \in S'$, (M',s') is a *refinement* of (M,s) (resp., (M,s) and (M',s') are *bisimilar*), written $(M,s) \succcurlyeq (M',s')$ (resp., $(M,s) \approx (M',s')$), if there is a refinement (resp., bisimulation) $\mathfrak{R}$ from M to M' s.t. $(s,s') \in \mathfrak{R}$. Note that $\succcurlyeq$ is a preorder (i.e., reflexive and transitive) and $\approx$ is an equivalence relation over pointed structures.

Remark 1. For each pointed structure (M,s), $(M,s) \approx Unw(M,s)$.

Refinement Modal Logic. We recall the syntax and semantics of one-agent *refinement modal logic* (RML) [18,4], an equally expressive extension of basic modal logic [3] obtained by adding the *existential and universal refinement quantifiers*. For technical convenience, the syntax of RML formulas φ over P is given in *positive form* as:

$$\varphi ::= p \mid \neg p \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \Diamond\varphi \mid \Box\varphi \mid \exists_r \varphi \mid \forall_r \varphi$$

where $p \in P$, $\Diamond\varphi$ reads as "possibly φ", $\Box\varphi$ reads as "necessarily φ", and $\exists_r$ and $\forall_r$ are the existential and universal refinement quantifiers. The *dual* $\widetilde{\varphi}$ of a RML formula φ is inductively defined as: $\widetilde{p} = \neg p$, $\widetilde{\neg p} = p$, $\widetilde{\varphi \vee \psi} = \widetilde{\varphi} \wedge \widetilde{\psi}$, $\widetilde{\Diamond\varphi} = \Box\widetilde{\varphi}$, $\widetilde{\Box\varphi} = \Diamond\widetilde{\varphi}$, $\widetilde{\exists_r \varphi} = \forall_r \widetilde{\varphi}$,

and $\widetilde{\forall_r\varphi} = \exists_r\widetilde{\varphi}$. We assume a standard DAG representation of a formula, hence, the size $|\varphi|$ of a formula φ is the number of vertices in the associated DAG (corresponding to the number of distinct subformulas of φ). For example $|p \vee p| = 2$ and $|p \vee q| = 3$. RML is interpreted over pointed structures (M,s). The satisfaction relation $(M,s) \models \varphi$ is inductively defined as follows (we omit the clauses for boolean connectives):

$$\begin{array}{ll}
(M,s) \models p & \texttt{iff} \quad p \in V(s) \text{ where } M = \langle S,E,V\rangle \\
(M,s) \models \Diamond\varphi & \texttt{iff} \quad \text{for some successor } t \text{ of } s \text{ in } M, (M,t) \models \varphi \\
(M,s) \models \Box\varphi & \texttt{iff} \quad \text{for all successors } t \text{ of } s \text{ in } M, (M,t) \models \varphi \\
(M,s) \models \exists_r\varphi & \texttt{iff} \quad \text{for some refinement } (M',s') \text{ of } (M,s), (M',s') \models \varphi \\
(M,s) \models \forall_r\varphi & \texttt{iff} \quad \text{for all refinements } (M',s') \text{ of } (M,s), (M',s') \models \varphi
\end{array}$$

Note that $(M,s) \models \varphi$ iff $(M,s) \not\models \widetilde{\varphi}$. If $(M,s) \models \varphi$, we say that (M,s) *satisfies* φ, or also that (M,s) is a *model* of φ. A RML formula φ is *satisfiable* if φ admits some model.

Fragments of RML. Let ML be the fragment of RML obtained by disallowing the refinement quantifiers, which corresponds to basic modal logic [3], and $\mathsf{RML}^{\forall}$ and $\mathsf{RML}^{\exists}$ be the fragments of RML obtained by disallowing the existential refinement quantifier and the universal refinement quantifier, respectively. Moreover, we introduce a family $\{\mathsf{RML}^k\}_{k\geq 1}$ of RML-fragments, where RML^k consists of the RML formulas whose *weak refinement quantifier alternation depth* (see Definition 1 below) is at most k.

Definition 1 (Weak Refinement Quantifier Alternation Depth). We first define the *weak alternation length* $\ell(\chi)$ of finite sequences $\chi \in \{\exists_r,\forall_r\}^*$ of refinement quantifiers: $\ell(\varepsilon) = 0$, $\ell(Q) = 1$ for every $Q \in \{\exists_r,\forall_r\}$, and $\ell(QQ'\chi)$ is $\ell(Q'\chi)$ if $Q = Q'$, and $\ell(Q'\chi)+1$ otherwise. For a RML formula φ, let $T(\varphi)$ be the standard tree encoding of φ, where each node is labeled by either a modality, or a boolean connective, or an atomic proposition. The *weak refinement quantifier alternation depth* $\Upsilon_w(\varphi)$ of a RML formula φ is the maximum of the alternation lengths $\ell(\chi)$ where χ is the sequence of refinement quantifiers along a path of $T(\exists_r\varphi)$ (note that we consider $T(\exists_r\varphi)$ and not $T(\varphi)$).

As an example, for $\varphi = (\forall_r\exists_r p) \vee \Box(\exists_r(p \wedge \forall_r q))$, $\Upsilon_w(\varphi) = 3$. Note that $\mathsf{RML}^{\exists} = \mathsf{RML}^1$ and $\mathsf{RML}^{\forall} \subseteq \mathsf{RML}^2$. Moreover, for each RML formula φ, $\Upsilon_w(\forall_r\varphi) = \Upsilon_w(\widetilde{\forall_r\varphi}) + 1$. The following example illustrates the succinctness of $\mathsf{RML}^{\exists}$ w.r.t. ML.

Example 1. For $n \geq 1$, an n-block is a sequence $b_1,\ldots,b_{n+1}$ of $n+1$ bits. The following $\mathsf{RML}^{\exists}$ formula φ_n is satisfied by a tree structure iff there are two paths from the root encoding two n-blocks of the form $b_1,\ldots,b_n,b_{n+1}$ and $b_1,\ldots,b_n,b'_{n+1}$ s.t. $b_{n+1} \neq b'_{n+1}$:

$$\varphi_n := \exists_r\big(\Diamond^{n+1}(0 \wedge \neg 1) \wedge \Diamond^{n+1}(1 \wedge \neg 0) \wedge \bigwedge_{i=1}^{n} \bigvee_{b\in\{0,1\}} \Box^i(b \wedge \neg(1-b))\big)$$

By using the approach in Section 6.2 of [4], one can easily show that any ML formula which is equivalent to φ_n has size singly exponential in n.

Investigated Problems. For each RML-fragment $\mathfrak{F}$, let $\mathsf{SAT}(\mathfrak{F})$ be the set of satisfiable $\mathfrak{F}$ formulas. In this paper, we investigate the complexity of $\mathsf{SAT}(\mathfrak{F})$ for any $\mathfrak{F} \in \{\mathsf{RML}, \mathsf{RML}^{\exists}, \mathsf{RML}^{\forall}, \mathsf{RML}^2, \ldots\}$. Figure 1 depicts our complexity results.

Assumption. Since RML is bisimulation invariant [18,4], by Remark 1, w.l.o.g. we can assume that the semantics of RML is restricted to tree structures.

Since RML and ML are equi-expressive [18,4], we easily obtain the following.

Proposition 1 (Finite Model Property). *Let* φ *be a* RML *formula and* $\langle T,V\rangle$ *be a tree structure satisfying* φ*. Then, there is a* finite refinement $\langle T_r,V_r\rangle$ *of* $\langle T,V\rangle$ *satisfying* φ*.*

3 Upper Bounds

In this section, we provide the upper bounds illustrated in Figure 1. Our approach consists of two steps. First, in Section 3.1, we show that RML enjoys a singly exponential size model property. Then, by using this result, we show in Section 3.2 that SAT(RML) can be decided by a singly exponential-time bounded ATM whose number of alternations on an input φ is at most $\Upsilon_w(\varphi)-1$ and whose initial state is existential. We fix a finite set P of atomic propositions and consider RML formulas and tree structures over P.

3.1 Exponential Size Model Property

In this section, we prove the following result.

Theorem 1 (Exponential Size Model Property). *For all satisfiable* RML *formulas* φ *and tree structures* $\langle T,V\rangle$ *such that* $\langle T,V\rangle$ *satisfies* φ*, the following holds: there exists a* finite refinement $\langle T',V'\rangle$ *of* $\langle T,V\rangle$ *such that* $\langle T',V'\rangle$ *satisfies* φ *and* $|T'|\leq|\varphi|^{3|\varphi|^2}$*.*

First, we summarize the main steps in the proof of Theorem 1. Given a RML formula φ, we associate with φ tableaux-based *finite* objects called *constraints systems for* φ (Definition 2). Essentially, a constraint system $\mathcal{S}$ for φ is a tuple of hierarchically ordered finite tree structures which intuitively represents an *extended model* of φ: (1) each node x in a tree structure of $\mathcal{S}$ is additionally labeled by a set of subformulas of φ which hold at the tree substructure rooted at node x, and, in particular, the first tree structure, called *main structure*, represents a model of φ, and (2) the other tree structures of $\mathcal{S}$ are used to manage the $\exists_r$-subformulas of φ. In fact, in order to be an *extended model* of φ, $\mathcal{S}$ has to satisfy additional structural requirements which capture the semantics of the boolean connectives and all the modalities except the universal refinement quantifier $\forall_r$, the latter being only semantically captured. Let $\mathcal{C}(\varphi)$ be the set of these constraints systems for φ, which are said to be *well-formed, saturated, and semantically* $\forall_r$*-consistent*. We individuate a subclass $\mathcal{C}_{min}(\varphi)$ of $\mathcal{C}(\varphi)$ consisting of '*minimal*' constraints systems for φ whose sizes are *singly exponential* in the size of φ, and which can be obtained from φ by applying structural *completion rules* (Definitions 3 and 4). Furthermore, we introduce a notion of 'refinement' between constraint systems for φ (Definition 5) which preserves the semantic $\forall_r$-consistency requirement. Then, given a *finite* tree structure $\langle T,V\rangle$ satisfying φ, we show that: (1) there is a constraint system $\mathcal{S}\in\mathcal{C}(\varphi)$ whose main structure is $\langle T,V\rangle$ (Lemma 1), and (2) starting from $\mathcal{S}$, it is possible to construct a minimal constraint system $\mathcal{S}_{min}\in\mathcal{C}_{min}(\varphi)$ which is a *refinement* of $\mathcal{S}$ (Lemma 3). This entails that the main structure of $\mathcal{S}_{min}$ is a refinement of $\langle T,V\rangle$ satisfying φ and having a single exponential size. Hence, by Proposition 1, Theorem 1 follows. Now, we proceed with the details of the proof of Theorem 1.

We denote by $\overline{P}$ the set of negations of propositions in P, i.e. $\overline{P}=\{\neg p \mid p\in P\}$. A set χ of RML formulas is *complete* if for each $p\in P$, either $p\in\chi$ or $\neg p\in\chi$. In the following, we fix a RML formula φ. The *closure* $\mathsf{cl}(\varphi)$ of φ is the set containing all the subformulas of φ and the formulas in $P\cup\overline{P}$. Moreover, $\mathsf{d}_{\Diamond,\Box}(\varphi)$ denotes the nesting depth of modalities $\Diamond$ and $\Box$ (in φ), and $\mathsf{d}_{\exists}(\varphi)$ denotes the nesting depth of modality $\exists_r$.

Definition 2. A *constraint system* for φ is a tuple $\mathcal{S} = \langle\langle T_1, L_1, \leftarrow_1\rangle, \ldots, \langle T_n, L_n, \leftarrow_n\rangle\rangle$, where for all $1 \leq i \leq n$, T_i is a *finite* tree and L_i and $\leftarrow_i$ are two T_i-labelings such that:

1. for each $x \in T_i$, $L_i(x)$ is a *complete* subset of $\mathsf{cl}(\varphi)$; moreover, $\varphi \in L_i(\varepsilon)$ if $i = 1$;
2. $\leftarrow_i : T_i \mapsto \{\bot\}$ if $i = 1$ ($\bot$ is for undefined), and $\leftarrow_i : T_i \mapsto \{j\} \times T_j$ for some $1 \leq j < i$ otherwise (note that $j < i$); moreover, for $i > 1$ and $x, x' \in T_i$ with $\leftarrow_i(x) = \langle j, y\rangle$ and $\leftarrow_i(x') = \langle j, y'\rangle$, if x' is a successor of x in T_i, then y' is a successor of y in T_j.

We denote by $\widetilde{L}_i$ the P-valuation over T_i defined as $\widetilde{L}_i(x) := L_i(x) \cap P$ for all $x \in T_i$, by $\mathcal{S}(i)$ the ith component of $\mathcal{S}$, i.e. $\langle T_i, L_i, \leftarrow_i\rangle$, and by $\mathsf{dim}(\mathcal{S})$ the number of $\mathcal{S}$ components, i.e., n. The (tree) structure $\langle T_1, \widetilde{L}_1\rangle$ is called the *main structure* of $\mathcal{S}$.

Let $1 \leq i, j \leq n$, $x \in T_i$, $y \in T_j$ and $\psi \in \mathsf{cl}(\varphi)$. We write $\langle j, y\rangle \leftarrow_{\mathcal{S}} \langle i, x\rangle$ to mean that $\leftarrow_i(x) = \langle j, y\rangle$, and $\mathcal{S} \vdash \langle i, x, \psi\rangle$ (resp., $\mathcal{S} \nvdash \langle i, x, \psi\rangle$) to mean that $\psi \in L_i(x)$ (resp., $\psi \notin L_i(x)$). If $\mathcal{S} \vdash \langle i, x, \psi\rangle$, we say that $\langle i, x, \psi\rangle$ is a *$\mathcal{S}$-constraint*. $\mathcal{S}$ contains a *clash* if $\mathcal{S} \vdash \langle i, x, p\rangle$ and $\mathcal{S} \vdash \langle i, x, \neg p\rangle$ for some $1 \leq i \leq n$, $x \in T_i$, and $p \in P$. Otherwise, $\mathcal{S}$ is called *clash-free*. Moreover, $\mathcal{S}$ is said to be *well-formed* if $\mathcal{S}$ is clash-free and whenever $\langle j, y\rangle \leftarrow_{\mathcal{S}} \langle i, x\rangle$, then $\widetilde{L}_j(y) = \widetilde{L}_i(x)$. Furthermore, $\mathcal{S}$ is said to be *semantically $\forall_r$-consistent* if whenever $\mathcal{S} \vdash (i, x, \forall_r \psi)$ then the tree structure $\langle T_i, \widetilde{L}_i\rangle_x$ satisfies $\forall_r \psi$.

If $\mathcal{S}$ is well-formed, then the labelings $\leftarrow_i$ induce a refinement hierarchy:

Remark 2. Let $\mathcal{S} = \langle\langle T_1, L_1, \leftarrow_1\rangle, \ldots, \langle T_n, L_n, \leftarrow_n\rangle\rangle$ be a *well-formed* constraint system for φ. Then, $\langle j, y\rangle \leftarrow_{\mathcal{S}} \langle i, x\rangle$ implies that $\langle T_i, \widetilde{L}_i\rangle_x$ is a refinement of $\langle T_j, \widetilde{L}_j\rangle_y$.

Definition 3 (Saturated Constraint Systems). A constraint system $\mathcal{S}$ for φ is *saturated* if none of the following *completion rules* are applicable to $\mathcal{S}$.

$\wedge$-rule: if $\mathcal{S} \vdash \langle i, x, \psi_1 \wedge \psi_2\rangle$, $\mathcal{S}(i) = \langle T, L, \leftarrow\rangle$, and $\{\psi_1, \psi_2\} \not\subseteq L(x)$
then *update* $L(x) := L(x) \cup \{\psi_1, \psi_2\}$

$\vee$-rule: if $\mathcal{S} \vdash \langle i, x, \psi_1 \vee \psi_2\rangle$, $\mathcal{S}(i) = \langle T, L, \leftarrow\rangle$, and $\{\psi_1, \psi_2\} \cap L(x) = \emptyset$
then *update* $L(x) := L(x) \cup \{\psi_k\}$ for some $k \in \{1, 2\}$

$\exists_r$-rule: if $\mathcal{S} \vdash \langle i, x, \exists_r \psi\rangle$, $\mathcal{S} := \langle\langle T_1, L_1, \leftarrow_1\rangle, \ldots, \langle T_n, L_n, \leftarrow_n\rangle\rangle$, and
$\mathcal{S} \nvdash \langle h, \varepsilon, \psi\rangle$ for each $h \leq \mathsf{dim}(\mathcal{S})$ such that $\leftarrow_h(\varepsilon) = \langle i, x\rangle$
then *update* $\mathcal{S} := \langle\langle T_1, L_1, \leftarrow_1\rangle, \ldots, \langle T_{n+1}, L_{n+1}, \leftarrow_{n+1}\rangle\rangle$, where
$T_{n+1} := \{\varepsilon\}$, $L_{n+1}(\varepsilon) := \{\psi\} \cup (L_i(x) \cap (P \cup \overline{P}))$, and $\leftarrow_{n+1}(\varepsilon) := \langle i, x\rangle$

$\Box$-rule: if $\mathcal{S} \vdash \langle i, x, \Box\psi\rangle$ and $\mathcal{S} \nvdash \langle i, x', \psi\rangle$ for some successor x' of x in $\mathcal{S}(i)$
then *let* $\mathcal{S}(i) = \langle T, L, \leftarrow\rangle$
update $L(x') := L(x') \cup \{\psi\}$ for each successor x' of x in T

$\Diamond$-rule: if $\mathcal{S} \vdash \langle i, x, \Diamond\psi\rangle$ and $\mathcal{S} \nvdash \langle i, x', \psi\rangle$ for each successor x' of x in $\mathcal{S}(i)$
then let $\langle i_0, x_0\rangle \leftarrow_{\mathcal{S}} \ldots \leftarrow_{\mathcal{S}} \langle i_k, x_k\rangle$ with $i_0 = 1$ and $\langle i_k, x_k\rangle = \langle i, x\rangle$
guess some *complete* set $\chi \subseteq P \cup \overline{P}$,
for each $q = k, k-1, \ldots, 0$ with $\mathcal{S}(i_q) = \langle T_q, L_q, \leftarrow_q\rangle$ do
update $T_q := T_q \cup \{x_q \cdot h_q\}$ for some $h_q \in \mathbb{N}$ such that $x_q \cdot h_q \notin T_q$
if $q < k$ then $L_q(x_q \cdot h_q) := \chi$ and $\leftarrow_{i_{q+1}}(x_{q+1} \cdot h_{q+1}) := \langle i_q, x_q \cdot h_q\rangle$
else $L_q(x_q \cdot h_q) := \{\psi\} \cup \chi$

Remark 3. Let $\mathcal{S}$ be a constraint system for φ. Then, applying any rule of Definition 3 to $\mathcal{S}$ yields a constraint system for φ.

The $\vee$-rule, the $\wedge$-rule, and the $\Box$-rule of Definition 3 are standard. The $\exists_r$-rule and the $\Diamond$-rule are the unique rules which add new nodes to the given constraint system $\mathcal{S}$ for φ. The $\exists_r$-rule is applicable to a $\mathcal{S}$-constraint $\xi = \langle i, x, \exists_r \psi \rangle$ if there are no $\mathcal{S}$-constraints (ξ-witnesses) of the form $\langle h, \varepsilon, \psi \rangle$ such that $\langle i, x \rangle \leftarrow_{\mathcal{S}} \langle h, \varepsilon \rangle$. The rule then adds a ξ-witness $\langle n+1, \varepsilon, \psi \rangle$ to $\mathcal{S}$ by extending $\mathcal{S}$ with a new component containing a single node (the root) whose label is propositionally consistent with the label of x. The $\Diamond$-rule is applicable to a $\mathcal{S}$-constraint $\xi = \langle i, x, \Diamond \psi \rangle$ if there are no $\mathcal{S}$-constraints (ξ-witnesses) of the form $\langle i, x', \psi \rangle$ where x' is a successor of x. Let $\langle i_0, x_0 \rangle \leftarrow_{\mathcal{S}} \ldots \leftarrow_{\mathcal{S}} \langle i_k, x_k \rangle$ be the maximal chain of 'backward links' from $\langle i_k, x_k \rangle = \langle i, x \rangle$. The rule then adds a ξ-witness $\langle i_k, x'_k, \psi \rangle$ to $\mathcal{S}$ (x'_k being a new successor of $x_k = x$), a complete set $\chi \subseteq P \cup \overline{P}$ is guessed, and the hierarchical structure of $\mathcal{S}$ is restored as follows: the rule adds the new constraints $\langle i_0, x'_0, \chi \rangle, \ldots, \langle i_k, x'_k, \chi \rangle$, where $x'_0, \ldots, x'_{k-1}$ are new successors of $x_0, \ldots, x_{k-1}$ respectively, and the new chain of 'backward links' $\langle i_0, x'_0 \rangle \leftarrow_{\mathcal{S}} \ldots \leftarrow_{\mathcal{S}} \langle i_k, x'_k \rangle$.

Lemma 1 (Soundness & Completeness). *Let $\langle T, V \rangle$ be a finite tree structure. Then, $\langle T, V \rangle$ satisfies* φ if and only if *there is a well-formed, saturated, and semantically $\forall_r$-consistent constraint system $\mathcal{S}$ for* φ *whose main structure is $\langle T, V \rangle$.*

Definition 4 (Minimal Constraint Systems). A constraint system $\mathcal{S}$ for φ is *initial* if $\mathcal{S} = \langle \langle \{\varepsilon\}, L, \leftarrow \rangle \rangle$, and for all $\psi \in L(\varepsilon)$, either $\psi = \varphi$ or $\psi \in P \cup \overline{P}$. A *minimal* constraint system $\mathcal{S}$ for φ is a constraint system for φ which can be obtained from some *initial* constraint system for φ by a sequence of applications of the rules of Definition 3.

The following lemma shows that every *minimal* constraint system for φ has a 'size' singly exponential in the size of φ.

Lemma 2. *Each* minimal *constraint system $\mathcal{S}$ for φ satisfies the following invariant: (i) each tree in $\mathcal{S}$ has height at most $d_{\Diamond,\Box}(\varphi)$ and branching degree at most $|\varphi|^{(2d_{\exists}(\varphi)+1)}$, and (ii) $\mathsf{dim}(\mathcal{S})$ is at most $|\varphi|^{4\cdot(d_{\exists}(\varphi))^2\cdot(d_{\Diamond,\Box}(\varphi)+1)}$. Moreover, any sequence of applications of the rules of Definition 3 starting from an* initial *constraint system for φ is finite.*

We introduce a notion of 'refinement' over constraint systems for φ, which generalizes the refinement preorder over finite structures. Moreover, this notion crucially preserves both well-formedness and the semantic $\forall_r$-consistency requirement (Lemma 3).

Definition 5 (Refinement for constraint systems). *Let $\mathcal{S} = \langle \langle T_1, L_1, \leftarrow_1 \rangle, \ldots, \langle T_n, L_n, \leftarrow_n \rangle \rangle$ and $\mathcal{S}' = \langle \langle T'_1, L'_1, \leftarrow'_1 \rangle, \ldots, \langle T'_m, L'_m, \leftarrow'_m \rangle \rangle$ be constraint systems for φ. $\mathcal{S}$ is a* refinement *of $\mathcal{S}'$ if there is a tuple $\mathcal{T} = \langle \uparrow_1, \ldots, \uparrow_n \rangle$ such that for all $1 \leq i \leq n$, there is $1 \leq j \leq m$ so that $\uparrow_i : T_i \mapsto T'_j$ and for all $x \in T_i$ with $\uparrow_i(x) = y$:*

1. *$L_i(x) \subseteq L'_j(y)$, $j = 1$ iff $i = 1$, and $y = \varepsilon$ iff $x = \varepsilon$;*
2. *for each successor x' of x in T_i, $\uparrow_i(x') = y'$ where y' is a successor of y in T'_j;*
3. *if $\leftarrow_i(x) = \langle i', x' \rangle$, then $\leftarrow'_j(y) = \langle j', y' \rangle$ and $\uparrow_{i'}(x') = y'$ (in particular, $\uparrow_{i'} : T_{i'} \mapsto T'_{j'}$).*

Lemma 3 (Minimalization). *Let $\mathcal{S}'$ be a constraint system for φ which is well-formed and semantically $\forall_r$-consistent. Then, any constraint system $\mathcal{S}$ for φ which is a refinement of $\mathcal{S}'$ is well-formed and semantically $\forall_r$-consistent too, and the main structure of $\mathcal{S}$ is a refinement of the main structure of $\mathcal{S}'$. Moreover, if $\mathcal{S}'$ is additionally saturated, there is a* minimal *and saturated constraint system $\mathcal{S}$ for φ which is a refinement of $\mathcal{S}'$.*

Sketched Proof. The first part of Lemma 3 follows from Definition 5 and the following crucial observation: if $\langle T_r, V_r\rangle$ is a refinement of a tree structure $\langle T, V\rangle$, then for each $\forall_r$-formula $\forall_r\psi$, $\langle T, V\rangle \models \forall_r\psi$ implies $\langle T_r, V_r\rangle \models \forall_r\psi$. For the second part of Lemma 3, let $\mathcal{S}'$ be a well-formed, saturated, and semantically $\forall_r$-consistent constraint system for φ. Since $\mathcal{S}'$ is well-formed, there is a unique *initial* constraint system $\mathcal{S}_0$ for φ s.t. $\mathcal{S}_0$ is a refinement of $\mathcal{S}'$. For each rule of Definition 3, we define an extension of such a rule which has the same *precondition* and the same *effect* (w.r.t. a given constraint system $\mathcal{S}$) with the difference that the nondeterministic choices are guided by $\mathcal{S}'$. By Lemma 2, it follows that any sequence of applications of the *new* rules starting from $\mathcal{S}_0$ is *finite*. Moreover, the application of these new rules preserves the property of a constraint system to be a refinement of $\mathcal{S}'$. Hence, we deduce that there is a *minimal* and saturated constraint system $\mathcal{S}$ for φ which is a refinement of $\mathcal{S}'$. □

Proof of Theorem 1. Let $\langle T, V\rangle$ be a tree structure satisfying φ. By Proposition 1, there is a finite refinement $\langle T_r, V_r\rangle$ of $\langle T, V\rangle$ satisfying φ. By Lemma 1, there is a well-formed, saturated, and semantically $\forall_r$-consistent constraint system $\mathcal{S}$ for φ whose main structure is $\langle T_r, V_r\rangle$. Thus, by Lemma 3, there is a *minimal*, well-formed, saturated, and semantically $\forall_r$-consistent constraint system $\mathcal{S}_{min}$ for φ whose main structure $\langle T_{min}, V_{min}\rangle$ is a refinement of $\langle T_r, V_r\rangle$. Hence, $\langle T_{min}, V_{min}\rangle$ is a refinement of $\langle T, V\rangle$ as well, and by Lemmata 1 and 2, $\langle T_{min}, V_{min}\rangle$ satisfies φ and $|T_{min}| \leq |\varphi|^{3|\varphi|^2}$. Hence, the result follows.

3.2 Checking Satisfiability

For a RML formula, the set $\mathsf{FL}(\varphi)$ of *first-level* subformulas of φ is defined as follows: if $\varphi = \varphi_1 \vee \varphi_2$ or $\varphi = \varphi_1 \wedge \varphi_2$, then $\mathsf{FL}(\varphi) = \mathsf{FL}(\varphi_1) \cup \mathsf{FL}(\varphi_2)$; otherwise $\mathsf{FL}(\varphi) = \{\varphi\}$.

Theorem 2. *$SAT(RML) \in AEXP_{pol}$ and $SAT(RML^k) \in \Sigma_k^{EXP}$ for each $k \geq 1$.*

Proof. For a RML formula φ, a *certificate* of φ is a finite tree structure $\langle T, V\rangle$ such that $|T| \leq |\varphi|^{3\cdot|\varphi|^2}$. Define:

$$\widehat{\mathsf{SAT}}(\mathsf{RML}) := \{(\varphi, \langle T, V\rangle) \mid \varphi \in \mathsf{RML} \textit{ and } \langle T, V\rangle \text{ is a certificate of } \varphi \text{ satisfying } \varphi\}$$

By Theorem 1, $\varphi \in \mathsf{SAT}(\mathsf{RML})$ iff $(\varphi, \langle T, V\rangle) \in \widehat{\mathsf{SAT}}(\mathsf{RML})$ for some certificate $\langle T, V\rangle$ of φ. Since $\langle T, V\rangle$ has size singly exponential in the size of φ, it suffices to show that $\widehat{\mathsf{SAT}}(\mathsf{RML})$ can be decided by a *polynomial-time* bounded ATM whose number of alternations on an input $(\varphi, \langle T, V\rangle)$ is at most $\Upsilon_w(\varphi) - 1$ and whose initial state is existential. For this, in turn, we show that $\widehat{\mathsf{SAT}}(\mathsf{RML})$ can be decided by a *nondeterministic polynomial-time bounded* procedure "*check*" that given an input $(\varphi, \langle T, V\rangle)$, uses in case $\Upsilon_w(\varphi) > 1$ as an *oracle* the same language $\widehat{\mathsf{SAT}}(\mathsf{RML})$ but with input queries of the form $(\psi, \langle T', V'\rangle)$, where $\Upsilon_w(\psi) < \Upsilon_w(\varphi)$ and $\psi \in \mathsf{cl}(\varphi)$. Hence, by standard arguments in complexity theory [11,13], the result follows. Procedure *check* is defined as follows.

check$(\varphi, \langle T, V\rangle)$ /** $\varphi \in \mathsf{RML}$ and $\langle T, V\rangle$ is a certificate of φ **/

$\mathcal{K} \leftarrow \{(\varphi, \langle T, V\rangle)\}$;
`while` $\mathcal{K} \neq \emptyset$ `do`
  `select` $(\psi, \langle T', V'\rangle) \in \mathcal{K}$; `update` $\mathcal{K} \leftarrow \mathcal{K} \setminus \{(\psi, \langle T', V'\rangle)\}$;
  `guess` $\mathcal{F} \subseteq \mathsf{FL}(\psi)$ and let $\psi_{\mathcal{F}}$ be the *boolean* formula obtained from ψ by replacing

each first-level subformula θ of ψ with `true` if $\theta \in \mathcal{F}$, and `false` otherwise;
`if` $\psi_{\mathcal{F}}$ evaluates to `false` `then` *reject the input*;
`for each` $\theta \in \mathcal{F}$ `do`
`case` $\theta = p$ with $p \in P$: `if` $p \notin V'(\varepsilon)$ `then` *reject the input*;
`case` $\theta = \neg p$ with $p \in P$: `if` $p \in V'(\varepsilon)$ `then` *reject the input*;
`case` $\theta = \Diamond\theta'$: `guess` a child x of the T'-root, `update` $\mathcal{K} \leftarrow \mathcal{K} \cup \{(\theta', \langle T', V' \rangle_x)\}$;
`case` $\theta = \Box\theta'$: `update` $\mathcal{K} \leftarrow \mathcal{K} \cup \{(\theta', \langle T', V' \rangle_{x_1}), \ldots, (\theta', \langle T', V' \rangle_{x_k})\}$
where $x_1, \ldots, x_k$ are the children of the root of T';
`case` $\theta = \exists_r\theta'$: `guess` a *certificate* $\langle T'', V'' \rangle$ of θ' which is a refinement of $\langle T', V' \rangle$ and `update` $\mathcal{K} \leftarrow \mathcal{K} \cup \{(\theta', \langle T'', V'' \rangle)\}$;[2]
`case` $\theta = \forall_r\theta'$: `query` the oracle for $\widetilde{\mathsf{SAT}}(\mathsf{RML})$ with input $(\widetilde{\forall_r\theta'}, \langle T', V' \rangle)$;
/** note that $\Upsilon_w(\widetilde{\forall_r\theta'}) < \Upsilon_w(\forall_r\theta') \leq \Upsilon_w(\varphi)$ **/
`if` the oracle answers *YES* `then` *reject the input*;
`end for`
`end while`
accept the input.

Correctness of the procedure *check* easily follows from Theorem 1. □

4 Lower Bounds

In this section, we provide the lower bounds illustrated in Figure 1. The main contribution is $\mathsf{AEXP}_{\mathsf{pol}}$-hardness of $\mathsf{SAT}(\mathsf{RML})$, which is proved by a polynomial-time reduction from a suitable $\mathsf{AEXP}_{\mathsf{pol}}$-complete problem. First, we define this problem.

Let $k \geq 1$. A *k-ary deterministic Turing Machine* is a deterministic Turing machine $\mathcal{M} = \langle k, I, A, Q, \{q_{acc}, q_{rej}\}, q_0, \delta \rangle$ operating on k ordered semi-infinite tapes and having just one read/write head, where: I (resp., $A \supset I$) is the input (resp., work) alphabet, A contains the blank symbol #, Q is the set of states, q_{acc} (resp., q_{rej}) is the terminal accepting (resp., rejecting) state, q_0 is the initial state, and $\delta : (Q \setminus \{q_{acc}, q_{rej}\}) \times A \to (Q \times A \times \{-1, +1\}) \cup \{1, \ldots, k\}$ is the transition function. In each non-terminal step, if the read/write head scans a cell of the ℓth tape ($1 \leq \ell \leq k$) and $(q, a) \in (Q \setminus \{q_{acc}, q_{rej}\}) \times A$ is the current pair state/ scanned cell content, the following occurs:

- $\delta(q, a) \in Q \times A \times \{-1, +1\}$ (*ordinary moves*): $\mathcal{M}$ overwrites the tape cell being scanned, there is a change of state, and the read/write head moves one position to the left (-1) or right (+1) in accordance with $\delta(q, a)$.
- $\delta(q, a) = h \in \{1, \ldots, k\}$ (*jump moves*): the read/write head jumps to the left-most cell of the hth tape and the state remains unchanged.

$\mathcal{M}$ *accepts* a k-ary input $(w_1, \ldots, w_k) \in (I^*)^k$, written $\mathcal{M}(w_1, \ldots, w_k)$, if the computation of $\mathcal{M}$ from $(w_1, \ldots, w_k)$ (initially, the ℓth tape contains the word w_ℓ, and the head points to the left-most cell of the first tape) is accepting. We consider the following problem.

[2] By Theorem 1, if there is a refinement of $\langle T', V' \rangle$ which satisfies θ', there is also a refinement of $\langle T', V' \rangle$ satisfying θ' which is a certificate of θ'. Moreover, checking for two given *finite* tree structures $\langle T, V \rangle$ and $\langle T', V' \rangle$, whether $\langle T, V \rangle$ is a refinement of $\langle T', V' \rangle$ (or, equivalently, $\langle T', V' \rangle$ is a simulation of $\langle T, V \rangle$) can be done in polynomial time (see, e.g., [12]).

Alternation Problem. An instance of the problem is a triple $(k,n,\mathcal{M})$, where $k \geq 1$, $n > 1$, and $\mathcal{M}$ is a *polynomial-time bounded* k-ary deterministic Turing Machine with input alphabet I. The instance $(k,n,\mathcal{M})$ is *positive* iff the following holds, where $\mathsf{Q}_\ell = \exists$ if ℓ is odd, and $\mathsf{Q}_\ell = \forall$ otherwise (for all $1 \leq \ell \leq k$),

$$\mathsf{Q}_1 x_1 \in I^{2^n}.\, \mathsf{Q}_2 x_2 \in I^{2^n}. \ldots \mathsf{Q}_k x_k \in I^{2^n}.\, \mathcal{M}(x_1,\ldots,x_k)$$

Note that the quantifications Q_i are restricted to words over I of length 2^n.

For $k \geq 1$, the *k-Alternation Problem* is the Alternation Problem restricted to instances of the form $(k,n,\mathcal{M})$ (i.e., the first input parameter is fixed to k), and the *Linear Alternation Problem* is the Alternation Problem restricted to instances of the form $(n,n,\mathcal{M})$.

Proposition 2. *The Linear Alternation Problem is AEXP_{pol}-complete and for all $k \geq 1$, the k-Alternation Problem is Σ_k^{EXP}-complete.*

The proof of Proposition 2 is standard. Fix an instance $(k,n,\mathcal{M})$ of the Alternation Problem with $\mathcal{M} = \langle k, I, A, Q, \{q_{acc}, q_{rej}\}, q_0, \delta\rangle$. Since $\mathcal{M}$ is polynomial-time bounded, there is an integer constant $c \geq 1$ such that when started on a k-ary input $(w_1,\ldots,w_k)$, $\mathcal{M}$ reaches a terminal configuration in at most $(|w_1| + \ldots + |w_k|)^c$ steps. A (k,n)-input is a k-ary input $(w_1,\ldots,w_k)$ such that $w_i \in I^{2^n}$ for all $1 \leq i \leq k$. Let $\mathsf{c}(k,n) := c \cdot (n + \lceil \log k \rceil)$, where $\lceil \log k \rceil$ denotes the smallest $i \in \mathbb{N}$ such that $i \geq \log k$. Note that a configuration of $\mathcal{M}$ reachable from a (k,n)-input, called (k,n)-*configuration*, can be described as a tuple $\overrightarrow{C} = (C_1,\ldots,C_k)$ of k words $C_1,\ldots,C_k$ over $A \cup (Q \times A)$ of length exactly $2^{\mathsf{c}(k,n)}$ such that for some $1 \leq \ell \leq k$, C_ℓ is of the form $w \cdot (q,a) \cdot w' \in A^* \times (Q \times A) \times A^*$, and for $i \neq \ell$, $C_i \in A^{2^{\mathsf{c}(k,n)}}$. For a (k,n)-input $(a \cdot w_1,\ldots,w_k)$, the associated *initial* (k,n)-*configuration* is $((q_0,a) \cdot w_1 \cdot \#^{2^{\mathsf{c}(k,n)}-2^n}, \ldots, w_k \cdot \#^{2^{\mathsf{c}(k,n)}-2^n})$. Thus, the computations of $\mathcal{M}$ from (k,n)-inputs, called (k,n)-*computations*, can be described by sequences π of at most $2^{\mathsf{c}(k,n)}$ (k,n)-configurations. In fact, w.l.o.g., we can assume that π has length *exactly* $2^{\mathsf{c}(k,n)}$. In the rest of this section, we prove the following result.

Theorem 3. *One can construct a RML^{k+1} formula φ in time polynomial in n, k, and the size of the TM $\mathcal{M}$ such that (i) φ is a $\mathsf{RML}^{\forall}$ formula if $k = 1$, and (ii) φ is satisfiable* if and only if *the instance $(k,n,\mathcal{M})$ of the Alternation Problem is positive.*

By Proposition 2 and Theorem 3, we obtain the following.

Corollary 1. *$\mathsf{SAT}(\mathsf{RML})$ is AEXP_{pol}-hard, $\mathsf{SAT}(\mathsf{RML}^{\forall})$ is NEXPTIME-hard, and for all $k \geq 1$, $\mathsf{SAT}(\mathsf{RML}^{k+1})$ is Σ_k^{EXP}-hard.*

Tree Encoding of (k,n)-Computations. In order to prove Theorem 3, first, we define an encoding of (k,n)-computations by suitable tree structures over P, where P is given by

$$P = \{0, 1, arg_1, \ldots, arg_k\} \cup \Lambda$$

and Λ is the set of triples (u_-, u, u_+) s.t. $u \in A \cup (Q \times A)$ and $u_-, u_+ \in A \cup (Q \times A) \cup \{\perp\}$ ($\perp$ is for undefined). An *extended TM block ext_bl* is a word over 2^P of length $2\mathsf{c}(k,n) + 2$ of the form $ext\_bl = \{bit_1\} \cdot \ldots \cdot \{bit_{\mathsf{c}(k,n)}\} \cdot bl$, where bl, called *TM block*, is of the form $bl = \{bit'_1\} \cdot \ldots \cdot \{bit'_{\mathsf{c}(k,n)}\} \cdot \{arg_\ell\} \cdot \{t\}$ with $1 \leq \ell \leq k$ and $t \in \Lambda$. The *content* $\mathsf{CON}(ext\_bl)$ (resp., $\mathsf{CON}(bl)$) of ext_bl (resp., bl) is bl (resp., t), the *component number* of ext_bl and bl is ℓ, and the *position number* of ext_bl (resp., bl) is the integer in

$[0, 2^{\mathsf{c}(k,n)} - 1]$ whose binary code is $bit_1, \ldots, bit_{\mathsf{c}(k,n)}$ (resp., $bit'_1, \ldots, bit'_{\mathsf{c}(k,n)}$). Intuitively, *ext_bl* encodes the triple $t = (C_\ell(i-1), C_\ell(i), C_\ell(i))$ with $i = \mathsf{ID}(bl)$ (where $C_\ell(i-1) = \perp$ if $i = 0$, and $C_\ell(i+1) = \perp$ if $i = 2^{\mathsf{c}(k,n)} - 1$) of the ℓth component C_ℓ of some (k,n)-configuration, the latter being the $(\mathsf{ID}(ext\_bl))$-th (k,n)-configuration of some (k,n)-computation. For a sequence $\pi = \overrightarrow{C}_0, \ldots, \overrightarrow{C}_{2^{\mathsf{c}(k,n)}-1}$ of $2^{\mathsf{c}(k,n)}$ (k,n)-configurations, we can encode π by the set $S_{ext\_bl}(\pi)$ of extended blocks defined as: $ext\_bl \in S_{ext\_bl}(\pi)$ iff there are $0 \leq i, j \leq 2^{\mathsf{c}(k,n)} - 1$ and $0 \leq \ell \leq k$ such that $\mathsf{ID}(ext\_bl) = i$, $\mathsf{CON}(ext\_bl) = bl$, $\mathsf{ID}(bl) = j$ and bl is the TM block associated with the jth symbol of the ℓth component of $\overrightarrow{C}_i$. The tree representation of the set $S_{ext\_bl}(\pi)$ is defined as follows.

Definition 6. A (k,n)-*computation tree code is* a tree structure $\langle T, V \rangle$ over P such that:

1. Each path of $\langle T, V \rangle$ from the root has length $2\mathsf{c}(k,n) + 2$, and each node is labeled exactly by a proposition in P. Moreover, $\langle T, V \rangle$ satisfies the ML-formula
$$\bigwedge_{i=1}^{2\mathsf{c}(k,n)} \Box^{i-1}((\Diamond 0 \wedge \Diamond 1) \wedge \Box(0 \vee 1)) \wedge \Box^{2\mathsf{c}(k,n)}(\bigwedge_{\ell=1}^{k} \Diamond arg_\ell \wedge \Box \bigvee_{\ell=1}^{k} arg_\ell) \wedge \Box^{2\mathsf{c}(k,n)+2} \bigvee_{t \in \Lambda} t$$
This requirement implies, in particular, that each path ν of $\langle T, V \rangle$ from the root is labeled by a word of the form $V(\varepsilon) \cdot ext\_bl$, where *ext_bl* is an extended TM block.
2. There is a sequence $\pi = \overrightarrow{C}_0, \ldots, \overrightarrow{C}_{2^{\mathsf{c}(k,n)}-1}$ of $2^{\mathsf{c}(k,n)}$ (k,n)-configurations such that the set of extended TM blocks of $\langle T, V \rangle$ corresponds to the set $S_{ext\_bl}(\pi)$.

We also need to encode existential and universal quantification on the different components of a (k,n)-input of the TM $\mathcal{M}$. This leads to the following definition.

Definition 7 (Initialized full (k,n)-computation tree codes). Let $1 \leq \ell \leq k$. A ℓ-*initialized full* (k,n)-*computation tree code* is a tree structure $\langle T, V \rangle$ over P such that:

1. *Fullness requirement.* $\langle T, V \rangle$ satisfies Property 1 of Definition 6. Moreover, let $\nu = z_0, \ldots, z_{2c(n)+1}, z_{2c(n)+2}$ be a path of $\langle T, V \rangle$ (from the root) encoding an extended TM block *ext_bl* with component number h such that either $h > \ell$ or $\mathsf{ID}(ext\_bl) > 0$. Then, for each $t \in \Lambda$, there is a child z of $z_{2c(n)+1}$ which is labeled by $\{t\}$.
2. ℓ-*initialization requirement*. There are $w_1, \ldots, w_\ell \in I^{2^n}$ s.t. for each extended block *ext_bl* of $\langle T, V \rangle$ with component number $1 \leq h \leq \ell$ and position number 0, $bl = \mathsf{CON}(ext\_bl)$ encodes the $\mathsf{ID}(bl)$th symbol of the hth component of any *initial* (k,n)-configuration associated with a (k,n)-input of the form $(w_1, \ldots, w_\ell, w'_{\ell+1}, \ldots, w'_k)$ for some $w'_{\ell+1}, \ldots, w'_k \in I^{2^n}$. We say that $w_1, \ldots, w_\ell \in I^{2^n}$ is the ℓ-ary input (which is uniquely determined) associated with $\langle T, V \rangle$ and we write $\langle T, V \rangle(w_1, \ldots, w_\ell)$.

Intuitively, a ℓ-initialized full (k,n)-computation tree code $\langle T, V \rangle$ associated with a ℓ-ary input $w_1, \ldots, w_\ell \in I^{2^n}$ encodes all the possible (k,n)-computations from (k,n)-inputs of the form $(w_1, \ldots, w_\ell, w'_{\ell+1}, \ldots, w'_k)$ for arbitrary words $w'_{\ell+1}, \ldots, w'_k \in I^{2^n}$. More precisely, by construction, the following holds.

Proposition 3. *Let $1 \leq \ell \leq k$, $w_1, \ldots, w_\ell \in I^{2^n}$, and $\langle T, V \rangle$ be a ℓ-initialized full (k,n)-computation tree code such that $\langle T, V \rangle(w_1, \ldots, w_\ell)$ holds. Then, the following holds:*

1. *case $\ell < k$: for each $w \in I^{2^n}$, there is a refinement $\langle T_r, V_r \rangle$ of $\langle T, V \rangle$ which is a $\ell + 1$-initialized full (k,n)-computation tree code satisfying $\langle T_r, V_r \rangle(w_1, \ldots, w_\ell, w)$. Moreover, for each refinement $\langle T_r, V_r \rangle$ of $\langle T, V \rangle$ which is a $\ell + 1$-initialized full (k,n)-computation tree code, there is $w \in I^{2^n}$ such that $\langle T_r, V_r \rangle(w_1, \ldots, w_\ell, w)$ holds.*

2. *case $\ell = k$: the set of refinements of $\langle T,V\rangle$ which are (k,n)-computation tree codes encoding (k,n)-computations is non-empty, and each of such refinements encodes the (k,n)-computation from the (k,n)-input $(w_1,\ldots,w_k)$.*

Lemma 4. *One can construct in time polynomial in n, k, and the size of the TM $\mathcal{M}$,*

1. *a $\mathsf{RML}^\forall$ formula φ^1_{init} over P such that given a tree structure $\langle T,V\rangle$, $\langle T,V\rangle$ satisfies φ^1_{init} if and only if $\langle T,V\rangle$ is a 1-initialized full (k,n)-computation tree code;*
2. *a $\mathsf{RML}^\forall$ formula φ^ℓ_{init} over P (for each $2 \le \ell \le k$) such that given a* refinement *$\langle T_r,V_r\rangle$ of a $\ell-1$-initialized full (k,n)-computation tree code, $\langle T_r,V_r\rangle$ satisfies φ^ℓ_{init}* if and only if *$\langle T_r,V_r\rangle$ is a ℓ-initialized full (k,n)-computation tree code;*
3. *a $\mathsf{RML}^\forall$ formula φ_{comp} over P such that given a* refinement *$\langle T_r,V_r\rangle$ of a k-initialized full (k,n)-computation tree code, $\langle T_r,V_r\rangle$ satisfies φ_{comp}* iff *$\langle T_r,V_r\rangle$ is a (k,n)-computation tree code encoding a (k,n)-computation;*
4. *a ML formula φ_{acc} over P such that given a (k,n)-computation tree code $\langle T,V\rangle$, $\langle T,V\rangle$ satisfies φ_{acc}* iff *the (k,n)-configuration with position number $2^{c(k,n)-1}$ encoded by $\langle T,V\rangle$ is accepting.*

Theorem 3 directly follows from the following two results (Theorems 4 and 5).

Theorem 4. *One can construct a RML^{k+1} formula φ in time polynomial in n, k, and the size of the TM $\mathcal{M}$ such that φ is satisfiable* if and only if

$$\mathsf{Q}_1 x_1 \in I^{2^n}.\, \mathsf{Q}_2 x_2 \in I^{2^n} \ldots \mathsf{Q}_k x_k \in I^{2^n}.\, \mathcal{M}(x_1,\ldots,x_k)$$

where $\mathsf{Q}_\ell = \exists$ if ℓ is odd, and $\mathsf{Q}_\ell = \forall$ otherwise (for all $1 \le \ell \le k$).

Proof. Let $\varphi^1_{init},\ldots,\varphi^k_{init}$, and φ_{comp} be the $\mathsf{RML}^\forall$ formulas satisfying Properties 1–3 of Lemma 4, and φ_{acc} be the ML formula satisfying Property 4 of Lemma 4. Then, the RML^{k+1} formula φ is defined as follows, where $\widetilde{\mathsf{Q}}_\ell = \exists_r$ and $\mathrm{op}_\ell = \wedge$ if ℓ is odd, and $\widetilde{\mathsf{Q}}_\ell = \forall_r$ and $\mathrm{op}_\ell = \rightarrow$ otherwise (for all $2 \le \ell \le k$):[3]

$$\varphi := \varphi^1_{init} \wedge \widetilde{\mathsf{Q}}_2(\varphi^2_{init}\ \mathrm{op}_2\ \widetilde{\mathsf{Q}}_3(\varphi^3_{init}\ \mathrm{op}_3\ \ldots\ \mathrm{op}_{k-1}\ \widetilde{\mathsf{Q}}_k(\varphi^k_{init}\ \mathrm{op}_k\ \widetilde{\mathsf{Q}}_k(\varphi_{comp}\ \mathrm{op}_k\ \varphi_{acc}))\ldots))$$

By construction and Lemma 4, it easily follows that φ is RML^{k+1} formula which can be constructed in time polynomial in n, k, and the size of the TM $\mathcal{M}$. Let $\varphi_1 := \varphi$, $\varphi_{k+1} := \widetilde{\mathsf{Q}}_k(\varphi_{comp}\ \mathrm{op}_k\ \varphi_{acc})$, and for each $2 \le \ell \le k$,

$$\varphi_\ell := \widetilde{\mathsf{Q}}_\ell(\varphi^\ell_{init}\ \mathrm{op}_\ell\ \widetilde{\mathsf{Q}}_{\ell+1}(\varphi^{\ell+1}_{init}\ \mathrm{op}_{\ell+1}\ \ldots\ \mathrm{op}_{k-1}\ \widetilde{\mathsf{Q}}_k(\varphi^k_{init}\ \mathrm{op}_k\ \widetilde{\mathsf{Q}}_k(\varphi_{comp}\ \mathrm{op}_k\ \varphi_{acc}))\ldots))$$

Correctness of the construction directly follows from the following claim, where a 0-initialized full (k,n)-computation tree code is an arbitrary tree structure.

Claim: let $0 \le \ell \le k, w_1,\ldots,w_\ell \in I^{2^n}$, and $\langle T,V\rangle$ be a ℓ-initialized full (k,n)-computation tree code such that $\langle T,V\rangle(w_1,\ldots,w_\ell)$ holds. Then, $\langle T,V\rangle$ satisfies $\varphi_{\ell+1}$ if and only if

$$\mathsf{Q}_{\ell+1} x_{\ell+1} \in I^{2^n} \ldots \mathsf{Q}_k x_k \in I^{2^n}.\, \mathcal{M}(w_1,\ldots,w_\ell,x_{\ell+1},\ldots,x_k)$$

The claim follows from Proposition 3 and Lemma 4. □

Theorem 5. *Let $k = 1$. Then, one can construct a $\mathsf{RML}^\forall$ formula $\varphi^\forall$ in time polynomial in n and the size of the TM $\mathcal{M}$ such that $\varphi^\forall$ is satisfiable* if and only if $\exists x \in I^{2^n}.\, \mathcal{M}(x)$.

[3] For RML formulas φ and ψ, $\varphi \rightarrow \psi$ is an abbreviation for $\widetilde{\varphi} \vee \psi$.

5 Concluding Remarks

An intriguing question left open is the complexity of satisfiability for *multi-agent* RML [18,4]. Our approach does not seem to scale to the multi-agent case. Indeed, in this more general setting, refinement is defined according to a designated agent so that refinement restricts (modulo bisimulation) the accessibility relation of the designated agent, but preserves (modulo bisimulation) those of all the other agents. From a technical point of view, this means that a generalization of the minimalization lemma for the multi-agent framework (Lemma 3) which preserves the crucial semantic $\forall_r$-consistency requirement does not seem possible, and a different more sophisticated approach may be required. Another interesting direction is to investigate the exact complexity of the fragments $\mathsf{RML}^{\exists}$, $\mathsf{RML}^{\forall}$, and RML^{k}, and the succinctness gap between RML^{k} and RML^{k+1} for each $k \geq 1$. Furthermore, since the modal μ-calculus extended with refinement quantifiers (RML^{μ}, for short) is non-elementarily decidable [4], it would be interesting to individuate weak forms of interactions between fixed-points and refinement quantifiers, which may lead to elementarily decidable and interesting RML^{μ}-fragments.

References

1. Alur, R., Henzinger, T.A., Kupferman, O.: Alternating-time temporal logic. Journal of the ACM 49(5), 672–713 (2002)
2. Balbiani, P., Baltag, A., van Ditmarsch, H., Herzig, A., Hoshi, T., De Lima, T.: 'Knowable' as 'known after an announcement'. Review of Symbolic Logic 1(3), 305–334 (2008)
3. Blackburn, P., de Rijke, M., Venema, Y.: Modal Logic. Cambridge Tracts in Theoretical Computer Science 53. Cambridge University Press, Cambridge (2001)
4. Bozzelli, L., van Ditmarsch, H., French, T., Hales, J., Pinchinat, S.: Refinement modal logic (2012), http://arxiv.org/abs/1202.3538
5. Chandra, A.K., Kozen, D.C., Stockmeyer, L.J.: Alternation. Journal of the ACM 28(1), 114–133 (1981)
6. Ferrante, J., Rackoff, C.: A decision procedure for the first order theory of real addition with order. SIAM Journal on Computing 4(1), 69–76 (1975)
7. Fine, K.: Propositional quantifiers in modal logic. Theoria 36(3), 336–346 (1970)
8. French, T.: Bisimulation quantifiers for modal logic. PhD thesis, University of Western Australia (2006)
9. Harel, D., Kozen, D., Tiuryn, J.: Dynamic Logic. MIT Press (2000)
10. Hollenberg, M.: Logic and bisimulation. PhD thesis, University of Utrecht (1998)
11. Johnson, D.S.: A catalog of complexity classes. In: Handbook of Theoretical Computer Science, pp. A:67–A:161. MIT Press (1990)
12. Kupferman, O., Vardi, M.Y.: Verification of Fair Transisiton Systems. In: Alur, R., Henzinger, T.A. (eds.) CAV 1996. LNCS, vol. 1102, pp. 372–382. Springer, Heidelberg (1996)
13. Papadimitriou, C.H.: Computational Complexity. Addison Wesley (1994)
14. Parikh, R., Moss, L., Steinsvold, C.: Topology and epistemic logic. In: Handbook of Spatial Logics, pp. 299–341. Springer (2007)
15. Pinchinat, S.: A Generic Constructive Solution for Concurrent Games with Expressive Constraints on Strategies. In: Namjoshi, K.S., Yoneda, T., Higashino, T., Okamura, Y. (eds.) ATVA 2007. LNCS, vol. 4762, pp. 253–267. Springer, Heidelberg (2007)

16. Rybina, T., Voronkov, A.: Upper Bounds for a Theory of Queues. In: Baeten, J.C.M., Lenstra, J.K., Parrow, J., Woeginger, G.J. (eds.) ICALP 2003. LNCS, vol. 2719, pp. 714–724. Springer, Heidelberg (2003)
17. van Ditmarsch, H., French, T.: Simulation and Information: Quantifying over Epistemic Events. In: Meyer, J.-J.C., Broersen, J. (eds.) KRAMAS 2008. LNCS, vol. 5605, pp. 51–65. Springer, Heidelberg (2009)
18. van Ditmarsch, H., French, T., Pinchinat, S.: Future event logic - axioms and complexity. In: Proc. 7th Advances in Modal Logic, vol. 8, pp. 77–99. College Publications (2010)
19. Visser, A.: Bisimulations, model descriptions and propositional quantifiers. Logic Group Preprint Series 161, Department of Philosophy, Utrecht University (1996)