The Complexity of One-Agent Refinement Modal Logic

Laura Bozzelli[a], Hans van Ditmarsch[b], Sophie Pinchinat[c]

[a] *Technical University of Madrid (UPM), Madrid, Spain, laura.bozzelli@fi.upm.es*
[b] *LORIA, CNRS / University of Lorraine, hans.van-ditmarsch@loria.fr*
[c] *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]

Keywords: modal logic, complexity of satisfiability, bisimulation quantifiers

1. Introduction

Modal logics with explicit or implicit propositional quantification. Refinement modal logic is a logic with propositional quantification. Modal logics augmented with propositional quantifiers, which allow to quantify over subsets of the domain of the current model, have been investigated since Fine's seminal paper [2]. Fine distinguishes three different propositional quantifications, which allow different kinds of model transformations: quantifying over propositionally definable subsets (over booleans), quantifying over subsets definable in the logical language (of basic modalities and quantifiers), and quantifying over all subsets. Only the first two are, in our modern terms, bisimulation preserving. Propositional quantification can easily lead to undecidable logics [2, 3]. Undecidability relies on the ability of propositional quantification to dictate the structural properties of the underlying model [3]. This has motivated, more recently, the introduction of bisimulation quantified logics [4, 5, 3, 6]. In that framework, the quantification is over the models which are bisimilar to the current model except for a propositional variable p. This operation is bisimulation preserving, and these logics are decidable.

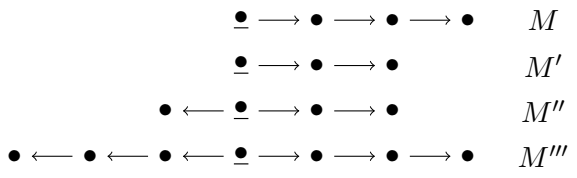
In [7] the authors propose a novel way of quantifying, namely over modally definable submodels. Unlike the above proposals, this not merely involves changing the valuation of a proposition in a subdomain, but *restricting the model* to that subdomain. The setting for these logics is how to quantify over information change. In the logic APAL of [7], an expression that we might

[1] This work is the revised and expanded version of [1].

write as $\exists\varphi$ for our purposes stands for 'there is a modal formula ψ such that in the submodel restriction to the states satisfying ψ it holds that φ'. This logic is undecidable [8].

Refinement modal logic (RML) [9, 10, 11] is a generalization of this perspective to more complex model transformations than submodel restrictions. This is achieved by existential and universal quantifiers which range over the *refinements* of the current model. In RML, an expression $\exists_r\varphi$ stands for 'there is a refinement wherein it holds that φ.' Given a model and a refinement of that model, we say that the two are in the *refinement relation*. From the *atoms/forth/back* requirements of bisimulation, a refinement (relation) between two given modal structures need only satisfy *atoms* and *back*. Refinement is therefore the dual of a simulation that need only satisfy *atoms* and *forth*, and it is more general than model restriction, since it is equivalent to bisimulation followed by model restriction. From a syntactic point of view, a *refinement formula* of the form $\exists_r\varphi$ can be mimicked by an existential bisimulation quantification followed by a *relativization* of φ, that is a simple syntactic transformation of φ which holds whenever φ holds in the outcome of this bisimulation quantification [11]. Just as in bisimulation quantified logics we have *explicit* quantification over propositional variables, refinement quantification as it is realized in refinement modal logic is *implicit* quantification over propositional variables, i.e., quantification over variables not occurring in the formula bound by the quantifier.

As an example of a refinement consider the following four rooted (underlined) structures.



With respect to the first model, M, the second one, M', is a model restriction. Model M'' is a refinement of M. It is not a model restriction. However, it is a model restriction of M''', a bisimilar copy of M. Refinements have really different properties, e.g., a formula like $\Diamond\Box\bot \wedge \Diamond\Diamond\Box\bot$ is clearly false in any model restriction of M, but it is true in its refinement M''. The root of the original model M satisfies the formula $\exists_r(\Diamond\Box\bot \wedge \Diamond\Diamond\Box\bot)$, where $\exists_r$ is the refinement quantifier.

As amply illustrated in [11], refinement quantification has applications in many settings: in logics for games [12, 6], it may correspond to a player discarding some moves; for program logics [13], it may correspond to operational refinement; and for logics for spatial reasoning, it may correspond to subspace projections [14].

Our contribution. We now get to the content of this paper and its novel contributions. We focus on complexity issues for (one-agent) refinement modal logic [9, 10, 11], the extension of (one-agent) basic modal logic (ML) obtained by adding the existential and universal refinement quantifiers $\exists_r$ and $\forall_r$.[2] It is known [10, 11] that RML has the same expressivity as ML, but the translation of RML into ML is of non-elementary complexity (see Section 6 in [10]) and no elementary upper bound is known for its satisfiability problem [11]. In fact, an upper bound in 2EXPTIME has been claimed in [10] by a tableaux-based procedure: the authors later concluded that the procedure is sound but not complete [11]. In this paper, our aim is to close that

[2]Refinement modal logic is called "Future Event Logic" in [10].

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 [15, 16]. 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 [15]. 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) [17]. 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 [18, 15, 19] such as the first-order theory of real addition with order [18, 15]. Formally, $\mathsf{AEXP}_{\mathsf{pol}}$ is the class of problems solvable by single exponential-time bounded ATM with a polynomial-bounded number of alternations.[3]

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. Note that our approach is completely different from the one proposed in [10]. There, a tableaux-based algorithm is given in a game-theoretic setting which is sound but not complete. The main reason of the incompleteness is that in the tableaux construction, the refinement quantification is just applied to model restrictions of the current "syntactical" model. In our approach instead, we use a tableaux construction only to infer a single exponential size model property. In particular, our tableaux construction does *not* represent a procedure for checking satisfiability of RML formulas: this is because the construction is guided by a "semantic" model, and crucially allows to deduce the existence of a *minimal* model (of size singly exponential in the size of the formula) which is a refinement of the given model.

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

Figure 1: Complexity results for satisfiability of RML and RML-fragments

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

[3] In Presburger arithmetic there are complexity classes of the form $\mathsf{STA}(f(n), g(n), h(n))$, where S is for Space, T for Time and A for Alternation, and where f , g, and h are functions. In that notation, $\mathsf{AEXP}_{\mathsf{pol}}$ is the class for any f (denoted by *), a g exponential in n and an h polynomial in n. See [20].

harder than satisfiability of ML, it is known [11] that RML is *doubly* exponentially more succinct than ML.

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 $\mathit{Unw}(M, s)$, 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 non-empty relation $\mathfrak{R} \subseteq S \times S'$ such that for all $(s, s') \in \mathfrak{R}$, the following holds:

- $V(s) = V'(s')$, and
- $(s', t') \in E'$ for some $t' \in S' \implies (s, t) \in E$ and $(t, t') \in \mathfrak{R}$ for some $t \in S$.

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), written $(M, s) \succcurlyeq (M', s')$, if there is a refinement $\mathfrak{R}$ from M to M' such that $(s, s') \in \mathfrak{R}$. Moreover, (M, s) and (M', s') are *bisimilar*, written $(M, s) \approx (M', s')$, if there is a bisimulation $\mathfrak{R}$ from M to M' such that $(s, s') \in \mathfrak{R}$. 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), $(M, s) \approx \mathit{Unw}(M, s)$.

Refinement Modal Logic. We recall the syntax and semantics of one-agent *refinement modal logic* (RML) [10, 11], an equally expressive extension of basic modal logic [21] 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 follows:

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

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; $\exists_r\varphi$ reads as "there is a refinement in which

φ", and dually for $\forall_r\varphi$.[4] The *dual* $\widetilde{\varphi}$ of an 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 $\widetilde{\forall_r\varphi} = \exists_r\widetilde{\varphi}$. For an RML formula φ, $\mathsf{d}_{\Diamond,\Box}(\varphi)$ denotes the nesting depth of modalities $\Diamond$ and $\Box$, and $\mathsf{d}_{\exists}(\varphi)$ denotes the nesting depth of modality $\exists_r$. As usual, the size $|\varphi|$ of an RML formula φ is the length of the string describing φ. Note that our complexity results continue to hold even if we assume a standard DAG representation of a formula φ (in this case, the size of a formula is defined as the number of distinct subformulas corresponding to the number of vertices of the DAG representation). 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):

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

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 *tree model of* φ is a tree structure $\langle T, V\rangle$ satisfying φ. An RML formula φ is *satisfiable* if φ has some model.

Fragments of RML. Let ML be the fragment of RML obtained by disallowing the refinement quantifiers, which corresponds to basic modal logic [21], 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 as follows: $\ell(\varepsilon) = 0$, $\ell(Q) = 1$ 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 an 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 an RML formula φ is the maximum over 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 $\mathsf{RML}^{\exists}$ with respect to 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*

[4] So, $\exists_r\varphi$ does *not* read as 'there is a formula φ': the variable of the quantifier $\exists_r$ is not the symbol following it, $\exists_r$ only implicitly quantifies over a variable. In later sections we will also use quantifiers $\exists$ and $\forall$ (without subscript 'r') in their normal usage, on a meta-level.

n-blocks of the form $b_1, \ldots, b_n, b_{n+1}$ *and* $b_1, \ldots, b_n, b'_{n+1}$ *such that* $b_{n+1} \neq b'_{n+1}$*:*

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

By using the approach in Section 6.2 of [11], 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 [10, 11], by Remark 1, without loss of generality we can assume that the semantics of RML is restricted to tree structures.

Since RML and ML are equi-expressive [10, 11], we easily obtain the following result.

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

Proof. Since for each RML formula, there is an equivalent ML formula [10, 11], it suffices to show that the result holds for ML. Let φ be a ML formula and $\langle T, V \rangle$ be a tree model of φ. Let $\langle T_{pr}, V_{pr} \rangle$ be the tree structure obtained from $\langle T, V \rangle$ by pruning all the subtrees rooted at nodes $x \in T \subseteq \mathbb{N}^*$ such that $|x|$ is strictly greater than the nesting depth $\mathsf{d}_{\Diamond,\Box}(\varphi)$ of modalities $\Diamond$ and $\Box$ in φ. By a straightforward induction on $\mathsf{d}_{\Diamond,\Box}(\varphi)$, we deduce that $\langle T_{pr}, V_{pr} \rangle$ satisfies φ as well. Now, we observe that $\langle T_{pr}, V_{pr} \rangle$ is a *refinement* of $\langle T, V \rangle$ having *finite* height (bounded by $\mathsf{d}_{\Diamond,\Box}(\varphi)$). Since $\langle T_{pr}, V_{pr} \rangle$ has finite height and the set P of atomic propositions labeling the nodes of T_{pr} is finite, we easily deduce that there is a *finite* tree structure $\langle T_r, V_r \rangle$ such that $\langle T_r, V_r \rangle$ and $\langle T_{pr}, V_{pr} \rangle$ are bisimilar. Hence, $\langle T_r, V_r \rangle$ is a refinement of $\langle T, V \rangle$. Moreover, since ML is bisimulation invariant, $\langle T_r, V_r \rangle$ satisfies φ as well, and we are done. □

Note that we can also give a direct proof of Proposition 1 which does not use the equi-expressiveness of ML and RML. This direct proof is by a simple structural induction on the size of the RML formula and for the induction step, the reasoning is similar to the one applied to ML in the proof above.

3. Exponential Size Model Property

In this section, we show that like the basic modal logic ML, RML enjoys a singly exponential size model property. More precisely, we prove the following result.

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

We fix a finite set P of atomic propositions and consider RML formulas and tree structures over P. First, we summarize the main steps in the proof of Theorem 2. Given an RML formula φ, we associate with φ tableaux-based *finite* objects called *constraint 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) 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 (more specifically, each of such tree structures is a model of some formula ψ with $\exists_r\psi$ being a subformula of φ). Moreover, 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. 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 constraint 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*" constraint 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 5 and 6). Furthermore, we introduce a notion of "refinement" between constraint systems for φ (Definition 7) 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 4), 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 11). 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 2 follows. Now, we proceed with the details of the proof of Theorem 2.

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 an RML formula φ. The *closure* $\mathsf{cl}(\varphi)$ of φ is the set containing all the subformulas of φ and the formulas in $P \cup \overline{P}$.

Definition 2 (Constraint Systems). *A* constraint system *for φ is a tuple $\mathcal{S}$ of the form $\mathcal{S} = \langle\langle T_1, L_1, \leftarrow_1\rangle, \ldots, \langle T_n, L_n, \leftarrow_n\rangle\rangle$, 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 satisfying the following:*

- *for each $x \in T_i$, $L_i(x)$ is a* complete *subset of $\mathsf{cl}(\varphi)$; moreover, $\varphi \in L_1(\varepsilon)$;*
- *$i = 1$: $\leftarrow_1$ is the mapping $\leftarrow_1 : T_1 \to \{\bot\}$ ($\bot$ is for undefined);*
- *$i > 1$: $\leftarrow_i$ is a mapping of the form $\leftarrow_i : T_i \to \{j\} \times T_j$ for some $1 \leq j < i$ such that for all $x, x' \in T_i$ with $\leftarrow_i(x) = \langle j, y\rangle$ and $\leftarrow_i(x') = \langle j, y'\rangle$, the following holds: 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 $\langle T_i, L_i, \leftarrow_i\rangle$ of $\mathcal{S}$, and by $\mathsf{dim}(\mathcal{S})$ the number n of $\mathcal{S}$ components. The tree structure $\langle T_1, \widetilde{L}_1\rangle$ is called the main structure *of $\mathcal{S}$.*

Fix a constraint system $\mathcal{S} = \langle\langle T_1, L_1, \leftarrow_1\rangle, \ldots, \langle T_n, L_n, \leftarrow_n\rangle\rangle$ for φ. 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 an $\mathcal{S}$*-constraint*.

Definition 3 (Well-Formed Constraint Systems). *The constraint system $\mathcal{S}$ 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 $\widetilde{L}_j(y) = \widetilde{L}_i(x)$.*

Definition 4 (Semantically $\forall_r$-consistent Constraint Systems). *The constraint system $\mathcal{S}$ 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 $\forall_r\psi$.*

Note that 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 5 (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 5 to $\mathcal{S}$ yields a constraint system for φ.

The $\vee$-rule, the $\wedge$-rule, and the $\Box$-rule of Definition 5 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 an $\mathcal{S}$-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 an $\mathcal{S}$-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" $\langle i_0, x'_0\rangle \leftarrow_{\mathcal{S}} \ldots \leftarrow_{\mathcal{S}} \langle i_k, x'_k\rangle$.

3.1. Soundness and Completeness

In this Subsection, we show that a *finite* tree structure $\langle T, V\rangle$ is a model of φ *iff* there is a well-formed, saturated, and semantically $\forall_r$-consistent constraint system $\mathcal{S}$ for φ whose main structure is $\langle T, V\rangle$. First, we prove the left implication.

Lemma 3 (Soundness). *Let $\mathcal{S} = \langle\langle T_1, L_1, \leftarrow_1\rangle, \ldots, \langle T_n, L_n, \leftarrow_n\rangle\rangle$ be a constraint system for φ such that $\mathcal{S}$ is well-formed, saturated, and semantically $\forall_r$-consistent. Then, the main structure of $\mathcal{S}$ satisfies φ.*

Proof. It suffices to show that whenever $\mathcal{S} \vdash \langle i, x, \psi\rangle$, then the tree structure $\langle T_i, \widetilde{L}_i\rangle_x$ satisfies ψ (recall that $\mathcal{S} \vdash \langle 1, \varepsilon, \varphi\rangle$). The proof is by structural induction on ψ:

- $\psi = p \in P$. Since $p \in L_i(x)$, $p \in \widetilde{L}_i(x)$ as well. Hence, the result follows.
- $\psi = \neg p$ and $p \in P$. Since $\neg p \in L_i(x)$ and $\mathcal{S}$ is well-formed (hence, clash-free), it holds that $p \notin L_i(x)$. Hence, $p \notin \widetilde{L}_i(x)$ and the result follows.
- $\psi = \psi_1 \wedge \psi_2$ (resp., $\psi = \psi_1 \vee \psi_2$). Since $\psi \in L_i(x)$ and $\mathcal{S}$ is saturated, by the precondition of the $\wedge$-rule (resp., $\vee$-rule), we obtain that $\mathcal{S} \vdash \langle i, x, \psi_1\rangle$ and $\mathcal{S} \vdash \langle i, x, \psi_2\rangle$ (resp., either $\mathcal{S} \vdash \langle i, x, \psi_1\rangle$ or $\mathcal{S} \vdash \langle i, x, \psi_2\rangle$). Hence, by the induction hypothesis, the result follows.
- $\psi = \Box\psi'$ (resp., $\psi = \Diamond\psi'$). Since $\psi \in L_i(x)$ and $\mathcal{S}$ is saturated, by the precondition of the $\Box$-rule (resp., $\Diamond$-rule), we obtain that $\mathcal{S} \vdash \langle i, x', \psi'\rangle$ for each (resp., for some) successor x' of x in T_i. Hence, by the induction hypothesis, the result follows.
- $\psi = \exists_r\psi'$. Since $\psi \in L_i(x)$ and $\mathcal{S}$ is saturated, by the precondition of the $\exists_r$-rule, there is some $h \leq \mathsf{dim}(\mathcal{S})$ such that $\langle i, x\rangle \leftarrow_{\mathcal{S}} \langle h, \varepsilon\rangle$ and $\mathcal{S} \vdash \langle h, \varepsilon, \psi'\rangle$. By the induction hypothesis, $\langle T_h, \widetilde{L}_h\rangle$ satisfies ψ'. Moreover, since $\langle i, x\rangle \leftarrow_{\mathcal{S}} \langle h, \varepsilon\rangle$ and $\mathcal{S}$ is well-formed, by Remark 2, it holds that $\langle T_h, \widetilde{L}_h\rangle$ is a refinement of $\langle T_i, \widetilde{L}_i\rangle_x$. Hence, $\langle T_i, \widetilde{L}_i\rangle_x$ satisfies $\exists_r\psi'$ and the result follows.
- $\psi = \forall_r\psi'$. Since $\psi \in L_i(x)$ and $\mathcal{S}$ is semantically $\forall_r$-consistent, the result follows.

This concludes the proof of the lemma. □

Now, we show that for every finite tree model $\langle T, V\rangle$ of φ, there is a well-formed, saturated, and semantically $\forall_r$-consistent constraint system $\mathcal{S}$ for φ whose main structure is $\langle T, V\rangle$. For this, we need additional definitions. For an RML formula ψ and a tree structure $\langle T, V\rangle$, the *ψ-completion* of $\langle T, V\rangle$ is the labeled tree $\langle T, L\rangle$, where for each $x \in T$, $L(x)$ is the set of formulas $\psi' \in \mathsf{cl}(\psi)$ such that $\langle T, V\rangle_x$ (the tree substructure of $\langle T, V\rangle$ rooted at x) satisfies ψ'. Note that $L(x)$ is a complete subset of $\mathsf{cl}(\varphi)$ and $L(x) \cap P = V(x)$. A constraint system $\mathcal{S} = \langle\langle T_1, L_1, \leftarrow_1\rangle, \ldots, \langle T_n, L_n, \leftarrow_n\rangle\rangle$ for φ is said to be an *extended model* for φ if for each $1 \leq i \leq n$, there is a formula $\psi_i \in \mathsf{cl}(\varphi)$ such that $\langle T_i, L_i\rangle$ is the ψ_i-completion of $\langle T_i, \widetilde{L}_i\rangle$ (note that since $\varphi \in L_1(\varepsilon)$, it holds that $\psi_1 = \varphi$).

Lemma 4 (Completeness). *Let $\langle T, V\rangle$ be a finite tree model of φ. Then, there is a well-formed, saturated, and semantically $\forall_r$-consistent constraint system $\mathcal{S}$ for φ whose main structure is $\langle T, V\rangle$.*

Proof. Let $\mathcal{S}_0 = \langle T, L, \leftarrow\rangle$, where $\langle T, L\rangle$ is the φ-completion of $\langle T, V\rangle$ and $\leftarrow(x) = \bot$ for each $x \in T$. Evidently, $\mathcal{S}_0$ is an extended model for φ which is well-formed and whose main structure is $\langle T, V\rangle$. We extend $\mathcal{S}_0$ by repeated applications of the following rule:

Semantic $\exists_r$-rule:

`if` $\mathcal{S} \vdash \langle i, x, \exists_r\psi\rangle$, $\mathcal{S} := \langle\langle T_1, L_1, \leftarrow_1\rangle, \ldots, \langle T_n, L_n, \leftarrow_n\rangle\rangle$ is an extended model for φ, 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
$\langle T_{n+1}, L_{n+1}\rangle$ is the ψ-completion of some *finite* refinement $\langle T_{n+1}, \widetilde{L}_{n+1}\rangle$ of $\langle T_i, \widetilde{L}_i\rangle_x$ satisfying ψ, and $\leftarrow_{n+1}$ is some labeling ensuring that $\leftarrow_{n+1}(\varepsilon) = \langle i, x\rangle$,
$\mathcal{S}$ is still a constraint system for φ, and $\langle i, z\rangle \leftarrow_{\mathcal{S}} \langle n+1, y\rangle$ implies $\widetilde{L}_{n+1}(y) = \widetilde{L}_i(z)$

Note that since $\exists_r\psi \in L_i(x)$ and $\mathcal{S} := \langle\langle T_1, L_1, \leftarrow_1\rangle, \ldots, \langle T_n, L_n, \leftarrow_n\rangle\rangle$ is an extended model of φ, there is a refinement $\langle T_{n+1}, V_{n+1}\rangle$ of $\langle T_i, \widetilde{L}_i\rangle_x$ satisfying ψ. Thus, since refinement is a preorder, by Proposition 1, there is a *finite* refinement $\langle T_{n+1}, \widetilde{L}_{n+1}\rangle$ of $\langle T_i, \widetilde{L}_i\rangle_x$ satisfying ψ. Hence, the semantic $\exists_r$-rule is well-defined. We show that by a finite number of applications of the semantic $\exists_r$-rule starting from the initial extended model $\mathcal{S}_0$ for φ (which is also well-formed), we obtain a well-formed, saturated, and semantically $\forall_r$-consistent constraint system for φ. Hence, since the main structure of $\mathcal{S}_0$ is $\langle T, V\rangle$ and applications of the semantic $\exists_r$-rule do not modify the main structure, the result follows. Note that if $\mathcal{S}$ is an extended model for φ, then $\mathcal{S}$ is semantically $\forall_r$-consistent, and at most the $\exists_r$-rule of Definition 5 is applicable to $\mathcal{S}$. Moreover, the application of the semantic $\exists_r$-rule preserves the property of a constraint system to be well-formed and an extended model for φ. Since the precondition of the semantic $\exists_r$-rule corresponds to the precondition of the $\exists_r$-rule in Definition 5, in order to complete the proof of the lemma, it suffices to show the following:

Claim. Any sequence of applications of the semantic $\exists_r$-rule starting from $\mathcal{S}_0$ is *finite*.

Proof of the claim. Fix an ordering $\varphi_1, \ldots, \varphi_k$ of the formulas in $\mathsf{cl}(\varphi)$ such that for all $i \neq j$, $\mathsf{d}_\exists(\varphi_i) > \mathsf{d}_\exists(\varphi_j)$ implies $i < j$. For each extended model $\mathcal{S}$ for φ obtained from $\mathcal{S}_0$ by a sequence of applications of the semantic $\exists_r$-rule, we associate to $\mathcal{S}$ a k-tuple of natural numbers $< \mathcal{S} >$ as follows. Let $\mathcal{S} = \langle\langle T_1, L_1, \leftarrow_1\rangle, \ldots, \langle T_n, L_n, \leftarrow_n\rangle\rangle$. For each $1 < i \leq \mathsf{dim}(\mathcal{S})$, let $\psi_i \in \mathsf{cl}(\varphi)$ be the formula such that $\exists_r\psi_i$ is in the precondition of the semantic $\exists_r$-rule instance whose application has generated the ith component $\mathcal{S}(i)$ of $\mathcal{S}$. Moreover, let $\psi_1 = \varphi$. Note that for all $1 \leq i \leq \mathsf{dim}(\mathcal{S})$, $\langle T_i, L_i\rangle$ is the ψ_i-completion of $\langle T_i, \widetilde{L}_i\rangle$. Then, for each $\psi \in \mathsf{cl}(\varphi)$, define

$$I(\mathcal{S}, \psi) = \{1 \leq i \leq dim(\mathcal{S}) \mid \psi_i = \psi\}$$

which represents the set of indices of the $\mathcal{S}$-components associated with the formula ψ. Moreover, for all $1 \leq i \leq \mathsf{dim}(\mathcal{S})$ and $\exists_r\psi \in \mathsf{cl}(\varphi)$, let $T(\mathcal{S}, i, \exists_r\psi)$ be the set of nodes $x \in T_i$ such that $\mathcal{S} \vdash (i, x, \exists_r\psi)$ and the *semantic* $\exists_r$-rule is applicable to the $\mathcal{S}$-constraint $\langle i, x, \exists_r\psi\rangle$. Then, $< S >= \langle n_1, \ldots, n_k\rangle$ where for all $1 \leq j \leq k$

$$n_j = \sum_{i \in I(\mathcal{S}, \varphi_j)} \sum_{\exists_r\psi \in cl(\varphi_j)} |T(\mathcal{S}, i, \exists_r\psi)|$$

Let $<_k$ be the lexicographic ordering over $\mathbb{N}^k$. Then, by construction, for each extended model $\mathcal{S}'$ for φ obtained from $\mathcal{S}$ by an application of the semantic $\exists_r$-rule, it easily follows that $< \mathcal{S}' > <_k < \mathcal{S} >$. Thus, since $<_k$ is well-founded, the claim follows. □

This concludes the proof of the lemma. □

3.2. Minimal Constraint Systems

Definition 6 (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 5.*

In the rest of this Subsection, we show that every *minimal* constraint system for φ has a "size" singly exponential in the size of φ. In particular, this entails that any sequence of applications of the rules of Definition 5 starting from an *initial* constraint system for φ is finite.

Fix a *minimal* constraint system $\mathcal{S} = \langle\langle T_1, L_1, \leftarrow_1\rangle, \ldots, \langle T_n, L_n, \leftarrow_n\rangle\rangle$ for φ. For all $1 \leq i \leq \mathsf{dim}(\mathcal{S})$ and $x \in T_i$, x satisfies one of the following:

- x is the root of T_i. Moreover, if $i > 1$, then the root of T_i is generated by an application of the $\exists_r$-rule. The formula ψ in the body of the $\exists_r$-rule which is added to the label of the root of T_i (when such a root is generated) is called *main formula* of $\mathcal{S}(i)$. The *main formula* of $\mathcal{S}(1)$ is φ.
- x is not the root of T_i, x is generated by executing the body of the $\Diamond$-rule, and in particular, x corresponds to node $x_k \cdot h_k$ in the body of the $\Diamond$-rule (see Definition 5).
- x is not the root of T_i, x is generated by executing the body of the $\Diamond$-rule, and in particular, x corresponds to some node $x_q \cdot h_q$ with $q < k$ in the body of the $\Diamond$-rule (see Definition 5).

In the first two cases, we say that x is a *main node*. Otherwise, x is said to be a *secondary node*. For each $1 \leq i \leq \mathsf{dim}(\mathcal{S})$, the main formula of $\mathcal{S}(i)$ is denoted by φ_i. Note that $\varphi_1 = \varphi$. For all $1 \leq i \leq \mathsf{dim}(\mathcal{S})$ and $x \in T_i$, define $\rightarrow_i(x) = \{1 \leq h \leq \mathsf{dim}(\mathcal{S}) \mid \leftarrow_h(\varepsilon) = \langle i, x\rangle\}$. Note that for each $h \in \rightarrow_i(x)$, $h > i$ and the root of T_h is generated by executing the $\exists_r$-rule applied to the constraint $\langle i, x, \exists_r\varphi_h\rangle$. Hence, in particular, $\exists_r\varphi_h \in L_i(x)$. First, we observe the following.

Proposition 5. *For the fixed minimal constraint system $\mathcal{S}$ for φ, let $1 \leq i \leq \mathsf{dim}(\mathcal{S})$, $x \in T_i$, and $h \in \rightarrow_i (x)$. Then, $L_i(x) \subseteq \mathsf{cl}(\varphi_i)$, $|\rightarrow_i(x)| \leq |\varphi_i|$, $\mathsf{d}_{\Diamond,\Box}(\varphi_h) \leq \mathsf{d}_{\Diamond,\Box}(\varphi_i)$, $\mathsf{d}_{\exists}(\varphi_h) < \mathsf{d}_{\exists}(\varphi_i)$, and φ_h is a subformula of φ_i.*

Proof. Let $1 \leq i \leq \mathsf{dim}(\mathcal{S})$ and $x \in T_i$. First, we prove that $L_i(x) \subseteq \mathsf{cl}(\varphi_i)$. Evidently, $L_i(\varepsilon) \subseteq \mathsf{cl}(\varphi_i)$. Thus, it suffices to show that for all nodes y and z of T_i such that z is a successor of y and $L_i(y) \subseteq \mathsf{cl}(\varphi_i)$, $L_i(z) \subseteq \mathsf{cl}(\varphi_i)$ holds as well. This follows from the fact that for each $\psi \in L_i(z)$ such that $\psi \notin P \cup \overline{P}$, either $\Diamond\psi \in L_i(y)$ or $\Box\psi \in L_i(y)$ (i.e., ψ has been added to the label of z by a $\Diamond$-rule or a $\Box$-rule), or ψ has been added to $L_i(z)$ by an application of the $\vee$-rule or $\wedge$-rule starting from a formula $\psi' \in L_i(z)$ such that ψ is a strict subformula of ψ'. Now,

let us show that $|{\rightarrow_i}(x)| \leq |\varphi_i|$. Let $k = |{\rightarrow_i}(x)|$. Since ${\rightarrow_i}(x)$ can be implicitly updated only by a $\exists_r$-rule application, by construction, ${\rightarrow_i}(x) = \{i_1, \ldots, i_k\}$, $\exists_r\varphi_{i_1}, \ldots, \exists_r\varphi_{i_k} \in L_i(x)$, and $\exists_r\varphi_{i_1}, \ldots, \exists_r\varphi_{i_k}$ are distinct formulas. Since $L_i(x) \subseteq \mathsf{cl}(\varphi_i)$, it follows that $\exists_r\varphi_{i_1}, \ldots, \exists_r\varphi_{i_k}$ are k distinct subformulas of φ. Hence, $k \leq |\varphi_i|$ and the result follows. Moreover, we obtain that φ_h is a subformula of φ_i, $\mathsf{d}_{\Diamond,\Box}(\varphi_h) \leq \mathsf{d}_{\Diamond,\Box}(\varphi_i)$, and $\mathsf{d}_{\exists}(\varphi_h) < \mathsf{d}_{\exists}(\varphi_i)$ for all $h \in {\rightarrow_i}(x)$, and we are done. □

Now, we give upper bounds on the heights of the trees of $\mathcal{S}$.

Proposition 6. *For the fixed minimal constraint system $\mathcal{S}$ for φ and for all $1 \leq i \leq \mathsf{dim}(\mathcal{S})$, the height of T_i is at most $\mathsf{d}_{\Diamond,\Box}(\varphi_i)$.*

Proof. For a finite set χ of RML formulas, define $\mathsf{d}_{\Diamond,\Box}(\chi) := max(\{\mathsf{d}_{\Diamond,\Box}(\psi) \mid \psi \in \chi\})$. The proof is by induction on $\mathsf{dim}(\mathcal{S}) - i$. We consider the induction step (the base case $i = \mathsf{dim}(\mathcal{S})$ being simpler). Thus, let $i < \mathsf{dim}(\mathcal{S})$. A node x of T_i is said to be *propositional* if $L_i(x) \subseteq P \cup \overline{P}$. Let $\pi = x_0, \ldots, x_k$ be a path of T_i from the root. We need to show that the length of π is at most $\mathsf{d}_{\Diamond,\Box}(\varphi_i)$. Observe that since $x_0 = \varepsilon$, $\varphi_i \in L_i(\varepsilon)$, and $L_i(\varepsilon) \subseteq \mathsf{cl}(\varphi_i)$ (Proposition 5), it holds that $\mathsf{d}_{\Diamond,\Box}(L_i(x_0)) = \mathsf{d}_{\Diamond,\Box}(\varphi_i)$. We distinguish two cases:

- $\pi = x_0 \ldots, x_k$ does *not* visit nodes which are both propositional and secondary. We show that for all $0 < j \leq k$, $\mathsf{d}_{\Diamond,\Box}(L_i(x_j)) < \mathsf{d}_{\Diamond,\Box}(L_i(x_{j-1}))$, hence, since $\mathsf{d}_{\Diamond,\Box}(L_i(x_0)) = \mathsf{d}_{\Diamond,\Box}(\varphi_i)$, the result follows. We assume the contrary and derive a contradiction. Then, there is $0 < j \leq k$ such that $\mathsf{d}_{\Diamond,\Box}(L_i(x_j)) \geq \mathsf{d}_{\Diamond,\Box}(L_i(x_{j-1}))$. There are two subcases:
  - x_j is a main node. Hence, there must be $\Diamond\psi \in L_i(x_{j-1})$ such that $\psi \in L_i(x_j)$. Moreover, for each $\psi' \in L_i(x_j) \setminus (P \cup \overline{P})$, either $\Diamond\psi' \in L_i(x_{j-1})$, or $\Box\psi' \in L_i(x_{j-1})$, or ψ' has been added to $L_i(x_j)$ by an application of the $\vee$-rule or $\wedge$-rule starting from a formula $\psi'' \in L_i(x_j)$ such that ψ' is a strict subformula of ψ''. It follows that $\mathsf{d}_{\Diamond,\Box}(L_i(x_j)) < \mathsf{d}_{\Diamond,\Box}(L_i(x_{j-1}))$, which is a contradiction.
  - x_j is a secondary node. By hypothesis, x_j is *not* a propositional node. Since when a secondary node is created, its label contains only atomic propositions, by the $\Box$-rule, there must be $\Box\psi \in L_i(x_{j-1})$ such that $\psi \in L_i(x_j)$. Moreover, for each $\psi' \in L_i(x_j) \setminus (P \cup \overline{P})$, either $\Box\psi' \in L_i(x_{j-1})$, or ψ' has been added to $L_i(x_j)$ by an application of the $\vee$-rule or $\wedge$-rule starting from a formula $\psi'' \in L_i(x_j)$ such that ψ' is a strict subformula of ψ''. It follows that $\mathsf{d}_{\Diamond,\Box}(L_i(x_j)) < \mathsf{d}_{\Diamond,\Box}(L_i(x_{j-1}))$, which is a contradiction.
- $\pi = x_0 \ldots, x_k$ visits some node which is both propositional and secondary. It easily follows that node x_k is both propositional and secondary. Hence, there is exactly one pair $\langle i', x'_k\rangle$ such that ${\leftarrow_{i'}}(x'_k) = \langle i, x_k\rangle$. Moreover, $x'_k \neq \varepsilon$. Let π' be the partial path of $T_{i'}$ from the root to node x'_k. Since $\mathcal{S}$ is a constraint system for φ, by Definition 2, there is $0 \leq q < k$ such that π' can be written in the form $\pi' = x'_q, x'_{q+1}, \ldots, x'_k$, where $x'_q = \varepsilon$ and ${\leftarrow_{i'}}(x'_l) = \langle i, x_l\rangle$ for all $q \leq l \leq k$. In particular, $i' \in {\rightarrow_i}(x_q)$ and $\exists_r\varphi_{i'} \in L_i(x_q)$. Since $i' > i$, by the induction hypothesis, the length of π' is at most $\mathsf{d}_{\Diamond,\Box}(\varphi_{i'})$. Since $\mathsf{d}_{\Diamond,\Box}(\varphi_{i'}) \leq \mathsf{d}_{\Diamond,\Box}(L_i(x_q))$ (because $\exists_r\varphi_{i'} \in L_i(x_q)$), we obtain that the suffix $x_q, \ldots, x_k$ of π has length bounded by $\mathsf{d}_{\Diamond,\Box}(L_i(x_q))$. Since the prefix $x_0, \ldots, x_q$ of π does not visit nodes which are both propositional and secondary (x_q is not propositional), by the first case, it holds that for all $0 < j \leq q$, $\mathsf{d}_{\Diamond,\Box}(L_i(x_j)) < \mathsf{d}_{\Diamond,\Box}(L_i(x_{j-1}))$. Hence, the length of π is at most $\mathsf{d}_{\Diamond,\Box}(L_i(x_0)) = \mathsf{d}_{\Diamond,\Box}(\varphi_i)$, and we are done.

Thus, in both cases the length of the path π of T_i is at most $\mathsf{d}_{\Diamond,\Box}(\varphi_i)$, which concludes the proof of the proposition. □

Next, we give upper bounds on the out-degrees of the nodes in the trees of $\mathcal{S}$.

Proposition 7. *For the fixed minimal constraint system $\mathcal{S}$ for φ, let $1 \leq i \leq \mathsf{dim}(\mathcal{S})$ and $x \in T_i$. Then, the out-degree of x in T_i (i.e., the number of successors of x in T_i) is at most $|\varphi_i|^{(2\mathsf{d}_\exists(\varphi_i)+1)}$.*

Proof. The proof is by induction on $\mathsf{dim}(\mathcal{S}) - i$. For the base case ($i = \mathsf{dim}(\mathcal{S})$), it holds that every node of T_i is a main node. Let $x_1, \ldots, x_k$ be the successors of x in T_i (which are created by applications of the $\Diamond$-rule). By the precondition of the $\Diamond$-rule, it follows that there are k distinct formulas $\Diamond\psi_1, \ldots, \Diamond\psi_k \in L_i(x)$. Since $L_i(x) \subseteq \mathsf{cl}(\varphi_i)$ (Proposition 5), $\Diamond\psi_1, \ldots, \Diamond\psi_k$ are k distinct subformulas of φ_i. Hence, the out-degree of x in T_i is bounded by $|\varphi_i|$. Thus, in this case the result holds. Now, assume that $i < \mathsf{dim}(\mathcal{S})$. Assume that $|\varphi_i| \geq 2$ (otherwise, the result is obvious) and x is a secondary node (the other case being simpler). Then, there is exactly one pair $\langle j, z\rangle$ such that $z \neq \varepsilon$ and $\leftarrow_j(z) = \langle i, x\rangle$ (in particular, x and z are created by the same application of the $\Diamond$-rule). Thus, since $\mathcal{S}$ is a constraint system, there is an ancestor x_a of x in T_i such that $\leftarrow_j(\varepsilon) = \langle i, x_a\rangle$, hence $j \in \rightarrow_i(x_a)$. Let $N_{main}(x)$ be the number of successors of x which are main nodes. By reasoning as in the base case, $N_{main}(x) \leq |\varphi_i|$. Then, by construction,

$$\text{out-degree of } x = N_{main}(x) + N_j(z) + \sum_{l \in \rightarrow_i(x)} N_l(\varepsilon)$$

where $N_j(z)$ is the out-degree of z in T_j, and for all $l \in \rightarrow_i(x)$, $N_l(\varepsilon)$ is the out-degree of the T_l-root. Since $j > i$ and $l > i$ for all $l \in \rightarrow_i(x)$, by the induction hypothesis we obtain the following (note that $\mathsf{d}_\exists(\varphi_i) > 1$)

$$\begin{aligned}
\text{out-degree of } x &\leq |\varphi_i| + |\varphi_j|^{(2\mathsf{d}_\exists(\varphi_j)+1)} + \sum_{l \in \rightarrow_i(x)} |\varphi_l|^{(2\mathsf{d}_\exists(\varphi_l)+1)} \\
&\leq |\varphi_i| + |\varphi_i|^{(2\mathsf{d}_\exists(\varphi_i)-1)} + \sum_{l \in \rightarrow_i(x)} |\varphi_i|^{(2\mathsf{d}_\exists(\varphi_i)-1)} \\
&\leq |\varphi_i| + |\varphi_i|^{(2\mathsf{d}_\exists(\varphi_i)-1)} + |\varphi_i| \cdot |\varphi_i|^{(2\mathsf{d}_\exists(\varphi_i)-1)} \\
&\leq 2 \cdot |\varphi_i| \cdot |\varphi_i|^{(2\mathsf{d}_\exists(\varphi_i)-1)} \\
&\leq |\varphi_i|^{(2\mathsf{d}_\exists(\varphi_i)+1)}
\end{aligned}$$

The second inequality directly follows from Proposition 5 and the fact that $j \in \rightarrow_i(x_a)$. The third inequality directly follows from Proposition 5, and the last two inequalities follows from the fact that $|\varphi_i| \geq 2$. This concludes the proof of the proposition. □

Finally, we give an upper bound on $\mathsf{dim}(\mathcal{S})$, the number of $\mathcal{S}$-components. For each $1 \leq i \leq \mathsf{dim}(\mathcal{S})$, let $H(\mathcal{S}, i)$ be the set of *i-descendants* in $\mathcal{S}$ defined as

$$H(\mathcal{S}, i) := \{i\} \cup \{j \in H(\mathcal{S}, h) \mid h \in \rightarrow_i(x) \text{ for some } x \in T_i\}$$

Note that $H(\mathcal{S}, i)$ is well defined since for all $x \in T_i$ and $h \in \rightarrow_i(x)$, $h > i$. Moreover, $\mathsf{dim}(\mathcal{S}) = |H(\mathcal{S}, 1)|$.

Proposition 8. *For the fixed minimal constraint system $\mathcal{S}$ for φ and for all $1 \leq i \leq \mathsf{dim}(\mathcal{S})$, it holds that $|H(\mathcal{S}, i)| \leq |\varphi_i|^{4\cdot(\mathsf{d}_\exists(\varphi_i))^2\cdot(\mathsf{d}_{\Diamond,\Box}(\varphi_i)+1)}$.*

Proof. The proof is by induction on $\mathsf{dim}(\mathcal{S}) - i$. For the base case ($i = \mathsf{dim}(\mathcal{S})$), $H(\mathcal{S}, i) = \{i\}$. Hence, the result follows. Now, assume that $i < \mathsf{dim}(\mathcal{S})$. If $\mathsf{d}_\exists(\varphi_i) = 0$, then $H(\mathcal{S}, i) = \{i\}$ and the result holds. Now, let $\mathsf{d}_\exists(\varphi_i) \geq 1$. Then:

$$
\begin{aligned}
|H(\mathcal{S}, i)| &= 1 + \sum_{x \in T_i, h \in \rightarrow_i(x)} |H(\mathcal{S}, h)| \\
&\leq 1 + \sum_{x \in T_i, h \in \rightarrow_i(x)} |\varphi_h|^{4\cdot(\mathsf{d}_\exists(\varphi_h))^2\cdot(\mathsf{d}_{\Diamond,\Box}(\varphi_h)+1)} \\
&\leq 1 + \sum_{x \in T_i, h \in \rightarrow_i(x)} |\varphi_i|^{4\cdot(\mathsf{d}_\exists(\varphi_i)-1)^2\cdot(\mathsf{d}_{\Diamond,\Box}(\varphi_i)+1)} \\
&\leq 1 + |\varphi_i| \cdot |\varphi_i|^{(2\mathsf{d}_\exists(\varphi_i)+1)\cdot\mathsf{d}_{\Diamond,\Box}(\varphi_i)} \cdot |\varphi_i|^{4\cdot(\mathsf{d}_\exists(\varphi_i)-1)^2\cdot(\mathsf{d}_{\Diamond,\Box}(\varphi_i)+1)} \\
&\leq 1 + |\varphi_i|^{(\mathsf{d}_{\Diamond,\Box}(\varphi_i)+1)\cdot(4\cdot(\mathsf{d}_\exists(\varphi_i))^2+5-6\mathsf{d}_\exists(\varphi_i))} \\
&\leq |\varphi_i|^{4\cdot(\mathsf{d}_\exists(\varphi_i))^2\cdot(\mathsf{d}_{\Diamond,\Box}(\varphi_i)+1)}
\end{aligned}
$$

The first inequality directly follows from the induction hypothesis and the second inequality directly follows from Proposition 5. For the third inequality, we use Propositions 5–7: by Propositions 6 and 7, $|T_i| \leq |\varphi_i|^{(2\mathsf{d}_\exists(\varphi_i)+1)\cdot\mathsf{d}_{\Diamond,\Box}(\varphi_i)}$, and by Proposition 5, for each $x \in T_i$, $|{\rightarrow_i}(x)| \leq |\varphi_i|$. Finally, the last inequality follows from the fact that $\mathsf{d}_\exists(\varphi_i) \geq 1$, hence, $|\varphi_i| > 1$. This concludes the proof of the proposition. □

Recall that the labeling of a node in a constraint system $\mathcal{S}$ for φ contains only formulas in $\mathsf{cl}(\varphi)$, and every application to $\mathcal{S}$ of a rule in Definition 5 either adds a new node or a new component with a single node, or adds a new formula in $\mathsf{cl}(\varphi)$ to a label of a node. Thus, since $\mathsf{cl}(\varphi)$ is finite and $\varphi_i \in \mathsf{cl}(\varphi)$ for each $1 \leq i \leq \mathsf{dim}(\mathcal{S})$, by Propositions 5–8, we obtain the main result of this Subsection.

Lemma 9. *Each* minimal *constraint system $\mathcal{S}$ for φ satisfies the following invariant: each tree in $\mathcal{S}$ has height at most $\mathsf{d}_{\Diamond,\Box}(\varphi)$ and branching degree at most $|\varphi|^{(2\mathsf{d}_\exists(\varphi)+1)}$, and $\mathsf{dim}(\mathcal{S})$ is at most $|\varphi|^{4\cdot(\mathsf{d}_\exists(\varphi))^2\cdot(\mathsf{d}_{\Diamond,\Box}(\varphi)+1)}$. Moreover, any sequence of applications of the rules of Definition 5 starting from an* initial *constraint system for φ is finite.*

3.3. Minimalization and proof of Theorem 2

In this subsection, 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 10). Then, we show that given a well-formed and saturated constraint system $\mathcal{S}$ for φ, it is possible to construct a saturated *minimal* constraint system $\mathcal{S}_{min}$ for φ which is a *refinement* of $\mathcal{S}$ (Lemma 11). Finally, these results and Lemmata 3, 4 and 9 provide a proof of Theorem 2.

Fix two constraint systems $\mathcal{S}$ and $\mathcal{S}'$ for φ with $\mathcal{S} = \langle\langle T_1, L_1, \leftarrow_1\rangle, \ldots, \langle T_n, L_n, \leftarrow_n\rangle\rangle$ and $\mathcal{S}' = \langle\langle T'_1, L'_1, \leftarrow'_1\rangle, \ldots, \langle T'_m, L'_m, \leftarrow'_m\rangle\rangle$.

Definition 7 (Refinement for constraint systems). *We say that $\mathcal{S}$ is a* refinement *of $\mathcal{S}'$ if there is a tuple $\mathcal{T} = \langle \mathcal{R}_1, \ldots, \mathcal{R}_n \rangle$ such that for all $1 \leq i \leq \mathsf{dim}(\mathcal{S})$, $\mathcal{R}_i$ is a mapping of the form $\mathcal{R}_i : T_i \mapsto T'_{i'}$ for some $1 \leq i' \leq \mathsf{dim}(\mathcal{S}')$, and for all $x \in T_i$, the following holds:*

1. *$L_i(x) \subseteq L'_{i'}(\mathcal{R}_i(x))$, $i' = 1$ iff $i = 1$, and $\mathcal{R}_i(x) = \varepsilon$ iff $x = \varepsilon$;*
2. *for each successor x' of x in T_i, $\mathcal{R}_i(x')$ is a successor of $\mathcal{R}_i(x)$ in $T'_{i'}$;*
3. *whenever $\langle j, y \rangle \leftarrow_{\mathcal{S}} \langle i, x \rangle$, then $\langle j', \mathcal{R}_j(y) \rangle \leftarrow_{\mathcal{S}'} \langle i', \mathcal{R}_i(x) \rangle$ and $\mathcal{R}_j : T_j \mapsto T'_{j'}$.*

We also say that $\mathcal{S}$ is a refinement of $\mathcal{S}'$ with respect to $\mathcal{T}$.

Intuitively, Properties 1–3 above ensure that if $\mathcal{S}'$ is additionally well-formed, then $\mathcal{S}$ is well-formed too and the subtree structures of $\mathcal{S}$ are refinements of the $\mathcal{T}$-associated subtree structures of $\mathcal{S}'$. Moreover, the *refinement hierarchy tree* of $\mathcal{S}$ (intuitively obtained by collapsing the trees in single nodes) is isomorphic to a subtree of the refinement hierarchy tree of $\mathcal{S}'$. In particular, the following holds.

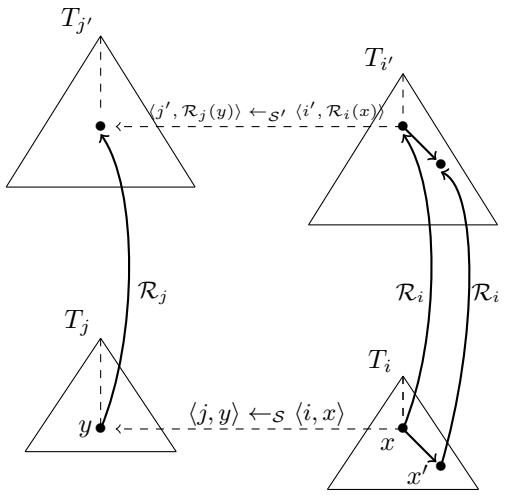


Figure 2: Illustration of Properties 2 and 3 in Definition 7

Lemma 10. *Assume that $\mathcal{S}$ is a refinement of $\mathcal{S}'$ with respect to $\mathcal{T} = \langle \mathcal{R}_1, \ldots, \mathcal{R}_n \rangle$ and $\mathcal{S}'$ is well-formed. Then, $\mathcal{S}$ is well-formed as well, and the main structure of $\mathcal{S}$ is a refinement of the main structure of $\mathcal{S}'$. Moreover, if $\mathcal{S}'$ is additionally semantically $\forall_r$-consistent, then $\mathcal{S}$ is semantically $\forall_r$-consistent too.*

Proof. Fix $1 \leq i \leq \mathsf{dim}(\mathcal{S})$ and let $\mathcal{R}_i : T_i \mapsto T'_{i'}$. First, we show that $\mathcal{S}$ is well-formed as well. By Property 1 of Definition 7, $L_i(x) \subseteq L'_{i'}(\mathcal{R}_i(x))$. Since $\mathcal{S}'$ is clash-free (because $\mathcal{S}'$ is well-formed) and i is arbitrary, it follows that $\mathcal{S}$ is clash-free as well. Moreover, since $L_i(x)$ is complete, we also obtain that $\widetilde{L}_i(x) = \widetilde{L}'_{i'}(\mathcal{R}_i(x))$. Now, assume that $\langle j, y \rangle \leftarrow_{\mathcal{S}} \langle i, x \rangle$. We show that $\widetilde{L}_i(x) = \widetilde{L}_j(y)$, hence $\mathcal{S}$ is well-formed. By Property 3 of Definition 7, $\langle j', \mathcal{R}_j(y) \rangle \leftarrow_{\mathcal{S}'} \langle i', \mathcal{R}_i(x) \rangle$ and $\mathcal{R}_j : T_j \mapsto T'_{j'}$. Then, we have that $\widetilde{L}_i(x) = \widetilde{L}'_{i'}(\mathcal{R}_i(x))$ and $\widetilde{L}_j(y) = \widetilde{L}'_{j'}(\mathcal{R}_j(y))$. Moreover, since $\mathcal{S}'$ is well-formed, $\widetilde{L}'_{i'}(\mathcal{R}_i(x)) = \widetilde{L}'_{j'}(\mathcal{R}_j(y))$. Hence, $\widetilde{L}_i(x) = \widetilde{L}_j(y)$ and the result follows. Moreover, by Property 2 of Definition 7, we obtain the following.

Claim. for all $x \in T_i$, $\langle T_i, \widetilde{L}_i\rangle_x$ is a refinement of $\langle T'_{i'}, \widetilde{L}'_{i'}\rangle_{\mathcal{R}_i(x)}$.

By Property 1 of Definition 7, $\mathcal{R}_1 : T_1 \mapsto T'_1$ and $\mathcal{R}_1(\varepsilon) = \varepsilon$. Thus, by the claim above, it follows that $\langle T_1, \widetilde{L}_1\rangle$ (the main structure of $\mathcal{S}$) is a refinement of $\langle T'_1, \widetilde{L}'_1\rangle$ (the main structure of $\mathcal{S}'$), which concludes the proof of the first part of the lemma. For the second part of the lemma, assume that $\mathcal{S}'$ is additionally semantically $\forall_r$-consistent. For the fixed $1 \leq i \leq \mathsf{dim}(\mathcal{S})$, let $x \in T_i$ and $\forall_r\psi \in L_i(x)$. We need to show that the tree structure $\langle T_i, \widetilde{L}_i\rangle_x$ satisfies $\forall_r\psi$. By Property 1 of Definition 7, $\forall_r\psi \in L'_{i'}(\mathcal{R}_i(x))$. Thus, since $\mathcal{S}'$ is semantically $\forall_r$-consistent, it holds that the tree structure $\langle T'_{i'}, \widetilde{L}'_{i'}\rangle_{\mathcal{R}_i(x)}$ satisfies $\forall_r\psi$. By the claim above, $\langle T_i, \widetilde{L}_i\rangle_x$ is a refinement of $\langle T'_{i'}, \widetilde{L}'_{i'}\rangle_{\mathcal{R}_i(x)}$. It follows that $\langle T_i, \widetilde{L}_i\rangle_x$ satisfies $\forall_r\psi$ as well (recall that refinement is a preorder), which concludes the proof of the lemma. □

Now, we can prove the following crucial lemma.

Lemma 11 (Minimalization). *Let $\mathcal{S}'$ be a constraint system for φ which is well-formed and saturated. Then, there exists a* minimal *and saturated constraint system $\mathcal{S}$ for φ which is a refinement of $\mathcal{S}'$.*

Proof. Let $\mathcal{S}' = \langle\langle T'_1, L'_1, \leftarrow'_1\rangle, \ldots, \langle T'_m, L'_m, \leftarrow'_m\rangle\rangle$ as in the statement of the lemma. First, for each rule of Definition 5, we define an extension of such a rule which has the same precondition and the same effect (with respect to a given constraint system $\mathcal{S}$ for φ) with the difference that the nondeterministic choices are guided by $\mathcal{S}'$. Moreover, these new rules are applicable to pairs $(\mathcal{S}, \mathcal{T})$ such that $\mathcal{S}$ is a refinement of $\mathcal{S}'$ with respect to $\mathcal{T}$ (*refinement invariant property*). These new rules are defined as follows, where for each rule, the parts which are not present in the corresponding rule of Definition 5 are underlined (for the rest, the two rules are identical). After having defined a new rule, we also prove that the rule is well-defined and its application preserves the refinement invariant property (correctness of the rule).

Assumption. $\mathcal{S}$ is a refinement of $\mathcal{S}'$ with respect to $\mathcal{T}$.

∧-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* $L(x) := L(x) \cup \{\psi_1, \psi_2\}$

Correctness of the ∧-rule. The new ∧-rule coincides with the ∧-rule of Definition 5 since in this case, there are not nondeterministic choices. Since $\mathcal{S}'$ is saturated, by Property 1 of Definition 7, it follows that the application of the rule preserves the refinement invariant property. □

∨-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 $k \in \{1, 2\}$
<u>such that $\mathcal{S}' \vdash \langle i', \mathcal{R}_i(x), \psi_k\rangle$ where $\mathcal{T} = \langle \mathcal{R}_1, \ldots, \mathcal{R}_n\rangle$ and $\mathcal{R}_i : T \mapsto T'_{i'}$</u>

Correctness of the ∨-rule. Like the ∨-rule of Definition 5, the new ∨-rule is applicable to an $\mathcal{S}$-constraint $\xi = \langle i, x, \psi_1 \vee \psi_2\rangle$ if there are no $\mathcal{S}$-constraints of the form $\langle i, x, \psi_k\rangle$ with $k \in \{1, 2\}$. Then, the rule adds a ξ-witness $\langle i, x, \psi_k\rangle$ for some $k \in \{1, 2\}$, with the new additional requirement that $\mathcal{S}' \vdash \langle i', \mathcal{R}_i(x), \psi_k\rangle$. Since $\mathcal{S}'$ is saturated and $L(x) \subseteq L'_{i'}(\mathcal{R}_i(x))$ (because $\mathcal{S}$ is a refinement of $\mathcal{S}'$ with respect to $\mathcal{T}$), such a k must exist. Hence, the rule is well-defined and, evidently, preserves the refinement invariant property. □

$\exists_r$**-rule:** `if` $\mathcal{S} \vdash \langle i, x, \exists_r \psi\rangle$, $\mathcal{S} := \langle\langle T_1, L_1, \leftarrow_1\rangle, \ldots, \langle T_n, L_n, \leftarrow_n\rangle\rangle, \underline{\mathcal{T} = \langle \mathcal{R}_1, \ldots, \mathcal{R}_n\rangle}$, 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* $\mathcal{S} := \langle\langle T_1, L_1, \leftarrow_1\rangle, \ldots, \langle T_{n+1}, L_{n+1}, \leftarrow_{n+1}\rangle\rangle$ and $\underline{\mathcal{T} = \langle \mathcal{R}_1, \ldots, \mathcal{R}_{n+1}\rangle}$,
where $T_{n+1} := \{\varepsilon\}$, $L_{n+1}(\varepsilon) := \{\psi\} \cup (L_i(x) \cap (P \cup \overline{P}))$, $\leftarrow_{n+1}(\varepsilon) := \langle i, x\rangle$,
$\mathcal{R}_{n+1} : T_{n+1} \mapsto T'_h$, $\mathcal{R}_{n+1}(\varepsilon) = \varepsilon$, where h satisfies:
$\underline{\langle i', \mathcal{R}_i(x)\rangle \leftarrow_{\mathcal{S}'} \langle h, \varepsilon\rangle,\ \mathcal{S}' \vdash \langle h, \varepsilon, \psi\rangle, \text{ and } \mathcal{R}_i : T_i \mapsto T'_{i'}}$

Correctness of the $\exists_r$-rule. Like the $\exists_r$-rule of Definition 5, the new $\exists_r$-rule is applicable to an $\mathcal{S}$-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 $\langle i, x\rangle \leftarrow_{\mathcal{S}} \langle h, \varepsilon\rangle$. Let $\mathcal{T} = \langle \mathcal{R}_1, \ldots, \mathcal{R}_n\rangle$ with $\mathcal{R}_i : T_i \mapsto T'_{i'}$. Since $\mathcal{S}$ is a refinement of $\mathcal{S}'$ with respect to $\mathcal{T}$, by Property 1 of Definition 7, $\langle i', \mathcal{R}_i(x), \exists_r \psi\rangle$ is a $\mathcal{S}'$-constraint. Thus, since $\mathcal{S}'$ is saturated, there must be a witness of $\langle i', \mathcal{R}_i(x), \exists_r \psi\rangle$ in $\mathcal{S}'$ of the form $\langle h, \varepsilon, \psi\rangle$ such that $\langle i', \mathcal{R}_i(x)\rangle \leftarrow_{\mathcal{S}'} \langle h, \varepsilon\rangle$. Then, as in Definition 5, the rule adds a ξ-witness $\langle n+1, \varepsilon, \psi\rangle$ to $\mathcal{S}$ by extending $\mathcal{S}$ with a new component $\langle T_{n+1}, L_{n+1}, \leftarrow_{n+1}\rangle$ containing a single node (the root) whose label is propositionally consistent with the label of x (i.e., $L_{n+1}(\varepsilon) := \{\psi\} \cup (L_i(x) \cap (P \cup \overline{P}))$). Additionally, the rule extends $\mathcal{T}$ by adding the new mapping $\mathcal{R}_{n+1} : T_{n+1} \mapsto T'_h$ with $\mathcal{R}_{n+1}(\varepsilon) = \varepsilon$. Since $\mathcal{S}'$ is well-formed, it holds that $L'_{i'}(\mathcal{R}_i(x)) \cap (P \cup \overline{P}) = L'_h(\varepsilon) \cap (P \cup \overline{P})$. Moreover, Property 1 of Definition 7 ensures that $L_i(x) \subseteq L'_{i'}(\mathcal{R}_i(x))$. Thus, since $L_{n+1}(\varepsilon) = \{\psi\} \cup (L_i(x) \cap (P \cup \overline{P}))$ and $\psi \in L'_h(\varepsilon)$, it holds that $L_{n+1}(\varepsilon) \subseteq L'_h(\varepsilon)$. It follows that after the updating, $\mathcal{S}$ is still a refinement of $\mathcal{S}'$ with respect to $\mathcal{T}$. □

$\Box$**-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

Correctness of the $\Box$-rule. The new $\Box$-rule coincides with the $\Box$-rule of Definition 5 since in this case, there are not nondeterministic choices. Since $\mathcal{S}'$ is saturated, by Properties 1 and 2 of Definition 7, it follows that the application of the rule preserves the refinement invariant property. □

$\Diamond$**-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 $\langle i_k, x_k\rangle = \langle i, x\rangle$
$\underline{\text{let } \langle j_0, \mathcal{R}_{i_0}(x_0)\rangle \leftarrow_{\mathcal{S}'} \ldots \leftarrow_{\mathcal{S}'} \langle j_k, \mathcal{R}_{i_k}(x_k)\rangle}$
$\underline{\text{where } \mathcal{T} = \langle \mathcal{R}_1, \ldots, \mathcal{R}_n\rangle,\ \mathcal{R}_{i_0} : T_{i_0} \mapsto T'_{j_0}, \ldots, \mathcal{R}_{i_k} : T_{i_k} \mapsto T'_{j_k}}$
$\underline{\text{let } y_k \text{ be a successor of } \mathcal{R}_{i_k}(x_k) \text{ in } T'_{j_k} \text{ such that } \psi \in L'_{j_k}(y_k)}$
$\underline{\text{let } \langle j_0, y_0\rangle \leftarrow_{\mathcal{S}'} \ldots \leftarrow_{\mathcal{S}'} \langle j_k, y_k\rangle}$
$\underline{\textit{guess} \text{ some } \textit{complete} \text{ set } \chi \subseteq P \cup \overline{P} \text{ such that } \chi = L'_{j_k}(y_k) \cap (P \cup \overline{P})}$,
`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 $x_q \cdot h_q \notin T_q$
$\underline{\mathcal{R}_{i_q}(x_q \cdot h_q) := y_q}$
`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` $L_q(x_q \cdot h_q) := \{\psi\} \cup \chi$

Correctness of the $\Diamond$-rule. Like the $\Diamond$-rule of Definition 5, the new $\Diamond$-rule is applicable to an $\mathcal{S}$-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 $\langle i_k, x_k\rangle = \langle i, x\rangle$. Note that $i_0 = 1$. Moreover, let $\mathcal{T} = \langle \mathcal{R}_1, \ldots, \mathcal{R}_n\rangle$. Then, since $\mathcal{S}$ is a refinement of $\mathcal{S}'$ with respect to $\mathcal{T}$, there is a maximal chain of "backward links" in $\mathcal{S}'$ of the

form $\langle j_0, \mathcal{R}_{i_0}(x_0)\rangle \leftarrow_{\mathcal{S}'} \ldots \leftarrow_{\mathcal{S}'} \langle j_k, \mathcal{R}_{i_k}(x_k)\rangle$, where $j_0 = 1$, $\mathcal{R}_{i_0} : T_{i_0} \mapsto T'_{j_0}$, ..., $\mathcal{R}_{i_k} : T_{i_k} \mapsto T'_{j_k}$. Moreover, $L_{i_k}(x_k) \subseteq L'_{j_k}(\mathcal{R}_{i_k}(x_k))$. Thus, since $\xi = \langle i, x, \Diamond\psi\rangle = \langle i_k, x_k, \Diamond\psi\rangle$ is an $\mathcal{S}$-constraint, $\langle j_k, \mathcal{R}_{i_k}(x_k), \Diamond\psi\rangle$ is a $\mathcal{S}'$-constraint, and because $\mathcal{S}'$ is saturated, there must be a witness of $\langle j_k, \mathcal{R}_{i_k}(x_k), \Diamond\psi\rangle$ in $\mathcal{S}'$ of the form $\langle j_k, y_k, \psi\rangle$ such that y_k is a child of $\mathcal{R}_{i_k}(x_k)$ in T'_{j_k}. Also note that the maximal chain of "backward links" in $\mathcal{S}'$ from $\langle j_k, \mathcal{R}_{i_k}(x_k)\rangle$ must be of the form $\langle j_0, y_0\rangle \leftarrow_{\mathcal{S}'} \ldots \leftarrow_{\mathcal{S}'} \langle j_k, y_k\rangle$ because $\mathcal{S}'$ is a constraint system for φ.

Then, like the $\Diamond$-rule of Definition 5, the rule adds a ξ-witness $\langle i_k, x'_k, \psi\rangle$ to $\mathcal{S}$ (x'_k being a new successor of $x_k = x$), and a complete set $\chi \subseteq P \cup \overline{P}$ is guessed with the new additional requirement that $\chi = L'_{j_k}(y_k) \cap (P \cup \overline{P})$. Moreover, 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" $\langle i_0, x'_0\rangle \leftarrow_{\mathcal{S}} \ldots \leftarrow_{\mathcal{S}} \langle i_k, x'_k\rangle$. Additionally, a $\mathcal{R}_{i_q}$-link from the new node x'_q to the $\mathcal{S}'$-node y_q is created for all $0 \leq q \leq k$. Since $\mathcal{S}'$ is well-formed and $\chi = L'_{j_k}(y_k) \cap (P \cup \overline{P})$, it holds that $\chi = L'_{j_k}(y_k) \cap (P \cup \overline{P}) = \ldots = L'_{j_0}(y_0) \cap (P \cup \overline{P})$. It follows that after the updating, $\mathcal{S}$ is still a refinement of $\mathcal{S}'$ with respect to $\mathcal{T}$. □

Since $\mathcal{S}'$ is well-formed, there is a unique *initial* constraint system $\mathcal{S}_0 = \langle\{\varepsilon\}, L, \leftarrow\rangle$ for φ such that $\mathcal{S}_0$ is a refinement of $\mathcal{S}'$. Let $\mathcal{T}_0 = \langle\mathcal{R}_1\rangle$ with $\mathcal{R}_1 : \{\varepsilon\} \mapsto \times\{\varepsilon\}$. Note that $\mathcal{S}_0$ is a refinement of $\mathcal{S}'$ with respect to $\mathcal{T}_0$. Now, every rule defined above has the same *precondition* and the same *effect* (with respect to a constraint system $\mathcal{S}$ for φ) of the corresponding rule in Definition 5. Moreover, we have proved that every rule defined above preserves the refinement invariant property. Hence, by Lemma 9, any sequence of applications of the rules defined above starting from $(\mathcal{S}_0, \mathcal{T}_0)$ is *finite* and leads to a pair $(\mathcal{S}, \mathcal{T})$ such that: (i) $\mathcal{S}$ is a refinement of $\mathcal{S}'$ with respect to $\mathcal{T}$, and (ii) $\mathcal{S}$ can be obtained from $\mathcal{S}_0$ by a sequence of applications of the rules of Definition 5. Hence, there is a *minimal* and saturated constraint system $\mathcal{S}$ for φ which is a refinement of $\mathcal{S}'$. This concludes the proof of the lemma. □

Now, we have all the material to prove Theorem 2.

Proof of Theorem 2. Let $\langle T, V\rangle$ be a tree model 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 4, 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 Lemmata 10 and 11, 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 3 and 9, $\langle T_{min}, V_{min}\rangle$ satisfies φ and $|T_{min}| \leq |\varphi|^{3|\varphi|^2}$. Hence, the result follows.

4. Upper bounds

In this section, we provide the upper bounds illustrated in Figure 1.

First, we briefly recall the framework of Alternating Turing Machines (ATM), see [17] for more details. An ATM $\mathcal{M}$ allows both existential choices (nondeterminism) and universal choices (parallelism). The set of $\mathcal{M}$-states is partitioned into a set of *existential* states and a set of *universal* states. This partition induces an analogous partition on the set of configurations in accordance with the associated state. The acceptance criterion of $\mathcal{M}$ can be defined via a reachability two-player turn-based game between player Eve and player Adam, where existential (resp., universal)

configurations are controlled by Eve (resp., Adam). A configuration C of $\mathcal{M}$ *leads to acceptance* iff there is a strategy of Eve from C which allows to select only computations (from C) ending in an accepting configuration. An input word α is *accepted* by $\mathcal{M}$ iff the initial configuration associated with α leads to acceptance. For technical convenience, we can also assume that the ATM $\mathcal{M}$ contains a set of *deterministic* states leading to deterministic configurations which have at most one successor. The ATM $\mathcal{M}$ is singly exponential-time bounded if there is an integer constant $c \geq 1$ such that for each input α, when started on α, no matter what are the universal and existential choices, $\mathcal{M}$ halts in at most $2^{|\alpha|^c}$ steps. The ATM $\mathcal{M}$ has a polynomial-bounded number of alternations if there is an integer constant $c \geq 1$ such that for all inputs α and computations π from α, the number of alternations of existential and universal configurations along π is at most $|\alpha|^c$.

The main result of this section is given by the following theorem, which is crucially based on Theorem 2. In fact, by using Theorem 2, it is a relatively simple task to construct a singly exponential-time bounded ATM accepting $\mathsf{SAT}(\mathsf{RML})$ with a number of alternations linear in the size of the given input φ. However, enforcing the number of alternations to match the refinement alternation depth $\Upsilon_w(\varphi)$ makes the construction a little more complicated.

Theorem 12. *One can construct a singly exponential-time bounded ATM accepting $\mathsf{SAT}(\mathsf{RML})$ whose number of alternations on an input $\varphi \in \mathsf{RML}$ is at most $\Upsilon_w(\varphi) - 1$ and whose initial state is existential.*

Proof. First, we need additional notation. Fix an RML formula φ. We denote by $\mathsf{SD}(\varphi)$ the set of formulas ψ such that either ψ is a subformula of φ or ψ is the dual of a subformula of φ. A *certificate* of φ is a finite tree structure $\langle T, V\rangle$ such that $|T| \leq |\varphi|^{3\cdot|\varphi|^2}$. A *well-formed set for* φ is a finite set $\mathcal{K}$ consisting of pairs $(\psi, \langle T, V\rangle)$ such that $\psi \in \mathsf{SD}(\varphi)$ and $\langle T, V\rangle$ is a certificate of ψ. We say that $\mathcal{K}$ is *universal* if each formula occurring in $\mathcal{K}$ is of the form $\forall_r\psi$ for some RML formula ψ.

We denote by $\Upsilon_w(\mathcal{K})$ the maximum over the alternation depths $\Upsilon_w(\psi)$, where ψ is a formula occurring in $\mathcal{K}$ (we set $\Upsilon_w(\mathcal{K}) = 0$ if $\mathcal{K} = \emptyset$), i.e. $\Upsilon_w(\mathcal{K}) = \max(\{0\} \cup \{\Upsilon_w(\psi) \mid \psi \in \mathcal{K}\})$. The *dual* of $\mathcal{K}$, written $\widetilde{\mathcal{K}}$ is the well-formed set for φ obtained by replacing each pair $(\psi, \langle T, V\rangle) \in \mathcal{K}$ with $(\widetilde{\psi}, \langle T, V\rangle)$. Recall that for each RML formula φ, $\Upsilon_w(\forall_r\varphi) = \Upsilon_w(\widetilde{\forall_r\varphi}) + 1$. It follows that for each non-empty *universal* well-formed set $\mathcal{K}$ for φ, $\Upsilon_w(\widetilde{\mathcal{K}}) = \Upsilon_w(\mathcal{K}) - 1$. A well-formed set $\mathcal{K}$ is *valid* if for each $(\psi, \langle T, V\rangle) \in \mathcal{K}$, $\langle T, V\rangle$ is a model of ψ.

Given an input φ, the ATM satisfying Theorem 12, that we call ATM *check*, uses two auxiliary ATM procedures *checkTrue* and *checkFalse*, illustrated in Figure 3 and in Figure 4 respectively, which take as input a well-formed set $\mathcal{K}$ for φ. Note that the procedure *checkFalse*($\mathcal{K}$) is the *dual* of *checkTrue*($\mathcal{K}$): it is simply obtained from *checkTrue*($\mathcal{K}$) by switching *accept* and *reject*, by switching existential choices and universal choices, and by converting the last call to *checkFalse* into *checkTrue*. Intuitively, the ATM *checkTrue*($\mathcal{K}$) accepts the input $\mathcal{K}$ iff $\mathcal{K}$ is valid. Dually, *checkFalse*($\mathcal{K}$) accepts the input $\mathcal{K}$ iff $\mathcal{K}$ is *not* valid. We assume that the set $\mathcal{K}$ is implemented by an ordered data structure in such a way that the selection of an element $(\psi, \langle T, V\rangle) \in \mathcal{K}$ where ψ is not of the form $\forall_r\psi'$ (if any) can be done deterministically (see the first line in the while loop of the procedures *checkTrue* and *checkFalse*).

Given an input $\varphi \in \mathsf{RML}$, the overall ATM *check* existentially chooses a certificate $\langle T, V\rangle$ of φ and calls *checkTrue* with input $\{(\varphi, \langle T, V\rangle)\}$.

checkTrue($\mathcal{K}$) /** $\mathcal{K}$ is a well-formed set for φ **/

`while` $\mathcal{K}$ is *not* universal `do`

`deterministically select` $(\psi, \langle T, V\rangle) \in \mathcal{K}$ such that ψ is not of the form $\forall_r \psi'$

`update` $\mathcal{K} \leftarrow \mathcal{K} \setminus \{(\psi, \langle T, V\rangle)\}$;

`case` $\psi = p$ with $p \in P$: `if` $p \notin V(\varepsilon)$ `then` *reject the input*;

`case` $\psi = \neg p$ with $p \in P$: `if` $p \in V(\varepsilon)$ `then` *reject the input*;

`case` $\psi = \psi_1 \vee \psi_2$: `existentially choose` $i = 1, 2$ and `update` $\mathcal{K} \leftarrow \mathcal{K} \cup \{(\psi_i, \langle T, V\rangle)\}$;

`case` $\psi = \psi_1 \wedge \psi_2$: `update` $\mathcal{K} \leftarrow \mathcal{K} \cup \{(\psi_1, \langle T, V\rangle), (\psi_2, \langle T, V\rangle)\}$;

`case` $\psi = \Diamond \psi'$: `existentially choose` a child x of the T-root,
`update` $\mathcal{K} \leftarrow \mathcal{K} \cup \{(\psi', \langle T, V\rangle_x)\}$;

`case` $\psi = \Box \psi'$: `update` $\mathcal{K} \leftarrow \mathcal{K} \cup \{(\psi', \langle T, V\rangle_{x_1}), \ldots, (\psi', \langle T, V\rangle_{x_k})\}$
where $x_1, \ldots, x_k$ are the children of the root of T;

`case` $\psi = \exists_r \psi'$: `existentially choose` a *certificate* $\langle T', V'\rangle$ of ψ' which is
a refinement of $\langle T, V\rangle$ and `update` $\mathcal{K} \leftarrow \mathcal{K} \cup \{(\psi', \langle T', V'\rangle)\}$;

`end while`

`if` $\mathcal{K} = \emptyset$ `then` *accept the input*

`else universally choose` $(\psi, \langle T, V\rangle) \in \widetilde{\mathcal{K}}$ and `call` *checkFalse*$(\{(\psi, \langle T, V\rangle)\})$

Figure 3: Procedure *checkTrue*($\mathcal{K}$)

Note that whenever there is a switch between the procedures *checkTrue* and *checkFalse*, e.g. from *checkTrue* to *checkFalse*, the input $\{(\psi, \langle T, V\rangle)\}$ of the latter is contained in the dual $\widetilde{\mathcal{K}}$ of a non-empty universal well-formed set $\mathcal{K}$ for φ, hence $\Upsilon_w(\{(\psi, \langle T, V\rangle)\}) < \Upsilon_w(\mathcal{K})$. Moreover, $\mathcal{K}$ contains only formulas ψ such that $\psi \in \mathsf{SD}(\theta)$ for a formula θ occurring in the former input. This ensures that the algorithm always terminates. Moreover, since the number of alternations of the ATM *check* between existential choices and universal choices is evidently the number of switches between the procedure calls *checkTrue* and *checkFalse*, and initially $\mathcal{K} = \{(\varphi, \langle T, V\rangle)\}$, we obtain the following result.

Claim 1. For the ATM *check*, the initial state is existential and the number of alternations on an input $\varphi \in \mathsf{RML}$ is at most $\Upsilon_w(\varphi) - 1$.

Now, we show that the ATM *check* runs in time singly exponential in the size of the input. Note that 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., [22]). Thus, since a certificate of a formula φ has size singly exponential in the size of φ, it follows that each case step of either procedure can be performed in time singly exponential in the size of φ. Hence, the result directly follows from the following claim.

Claim 2. Given an input $\varphi \in \mathsf{RML}$, the number of iterations of the while loop in each call of the procedure *checkTrue* (resp., *checkFalse*) is singly exponential in the size of φ.

Proof of Claim 2. Let $T(\varphi)$ be the standard tree encoding of φ, where each node is labeled by some subformula of φ, and define $C_\varphi := |\varphi|^{3 \cdot |\varphi|^2}$ (i.e., the size of a certificate of φ). Let $\psi \in \mathsf{SD}(\varphi)$. If ψ is a subformula of φ, we define d_ψ as the maximum over the distances from the root in $T(\varphi)$ of ψ-labeled nodes. If instead ψ is the dual of a subformula of φ, we let $d_\psi := d_{\widetilde{\psi}}$. Then, in order

checkFalse($\mathcal{K}$) /** $\mathcal{K}$ is a well-formed set for φ **/

`while` $\mathcal{K}$ is *not* universal `do`

`deterministically select` $(\psi, \langle T, V\rangle) \in \mathcal{K}$ such that ψ is not of the form $\forall_r \psi'$

`update` $\mathcal{K} \leftarrow \mathcal{K} \setminus \{(\psi, \langle T, V\rangle)\}$;

`case` $\psi = p$ with $p \in P$: `if` $p \notin V(\varepsilon)$ `then` *accept the input*;

`case` $\psi = \neg p$ with $p \in P$: `if` $p \in V(\varepsilon)$ `then` *accept the input*;

`case` $\psi = \psi_1 \vee \psi_2$: `universally choose` $i = 1, 2$ and `update` $\mathcal{K} \leftarrow \mathcal{K} \cup \{(\psi_i, \langle T, V\rangle)\}$;

`case` $\psi = \psi_1 \wedge \psi_2$: `update` $\mathcal{K} \leftarrow \mathcal{K} \cup \{(\psi_1, \langle T, V\rangle), (\psi_2, \langle T, V\rangle)\}$;

`case` $\psi = \Diamond\psi'$: `universally choose` a child x of the T-root,
`update` $\mathcal{K} \leftarrow \mathcal{K} \cup \{(\psi', \langle T, V\rangle_x)\}$;

`case` $\psi = \Box\psi'$: `update` $\mathcal{K} \leftarrow \mathcal{K} \cup \{(\psi', \langle T, V\rangle_{x_1}), \ldots, (\psi', \langle T, V\rangle_{x_k})\}$
where $x_1, \ldots, x_k$ are the children of the root of T;

`case` $\psi = \exists_r \psi'$: `universally choose` a *certificate* $\langle T', V'\rangle$ of ψ' which is
a refinement of $\langle T, V\rangle$ and `update` $\mathcal{K} \leftarrow \mathcal{K} \cup \{(\psi', \langle T', V'\rangle)\}$;

`end while`

`if` $\mathcal{K} = \emptyset$ `then` *reject the input*

`else existentially choose` $(\psi, \langle T, V\rangle) \in \widetilde{\mathcal{K}}$ and `call` *checkTrue*($\{(\psi, \langle T, V\rangle)\}$)

Figure 4: Procedure *checkFalse*($\mathcal{K}$)

to prove the claim, it suffices to show that for all computations π of the ATM *check* from input φ, the overall number N_ψ of iterations of the while loops (of procedures *checkTrue* and *checkFalse*) along π where formula ψ is processed, is at most $(|\varphi| \cdot C_\varphi)^{d_\psi}$. The proof is done by induction on d_ψ. For the base case, $d_\psi = 0$. Therefore, $\psi = \varphi$ or $\psi = \widetilde{\varphi}$, and by construction of the algorithm, N_φ and $N_{\widetilde{\varphi}}$ are at most equal to 1. Hence, the result holds. For the induction step, assume that $d_\psi > 0$. We consider the case where ψ is a subformula of φ (the case where $\widetilde{\psi}$ is a subformula of φ is similar). Then, the result follows from the following chain of inequalities, where $P(\psi)$ denotes the set of subformulas labeling the nodes of $T(\varphi)$ which are parents of the nodes labeled by ψ.

$$N_\psi \leq \sum_{\psi_0 \in P(\psi)} N_{\psi_0} \cdot C_\varphi \leq \sum_{\psi_0 \in P(\psi)} (|\varphi| \cdot C_\varphi)^{d_{\psi_0}} \cdot C_\varphi \leq (|\varphi| \cdot C_\varphi)^{d_\psi}$$

The first inequality directly follows from the construction of the algorithm (note that if $\psi_0 = \Box\psi$, then the processing of ψ_0 in an iteration of the two while loops generates at most C_φ new "copies" of ψ). The second inequality follows from the induction hypothesis and the last inequality follows from the fact that $|P(\psi)| \leq |\varphi|$ and $d_{\psi_0} \leq d_\psi - 1$ for all $\psi_0 \in P(\psi)$. This concludes the proof of Claim 2. □

It remains to show that the ATM *check* accepts $\mathsf{SAT}(\mathsf{RML})$. The proof is crucially based on Theorem 2. Fix an input φ. Evidently, after the first call to *checkTrue*, each configuration of the procedure *check* can be described by a triple $(\ell, \mathcal{K}, f)$, where: (i) $\mathcal{K}$ is a well-formed set for φ, (ii) $f =$ `true` if $\mathcal{K}$ is processed within *checkTrue*, and $f =$ `false` otherwise, and (iii) ℓ is an instruction label corresponding to one of the instructions of the procedures *checkTrue* and *checkFalse*. We denote by ℓ_0 the label associated with the while instruction. A *main configuration* is a configuration associated with the label ℓ_0. By construction of the algorithm, the ATM *check* accepts φ

iff there exists a certificate $\langle T, V\rangle$ of φ such that the main configuration $(\ell_0, \{(\varphi, \langle T, V\rangle)\}, \texttt{true})$ leads to acceptance. By Proposition 1 and Theorem 2, φ is satisfiable *iff* there exists a certificate $\langle T, V\rangle$ of φ which is a model of φ. Hence, in order to conclude the proof of Theorem 12, it suffices to prove the following.

Claim 3. Let $\mathcal{K}$ be a well-formed set for φ. Then, the following holds:

1. the main configuration $(\ell_0, \mathcal{K}, \texttt{true})$ leads to acceptance *iff* $\mathcal{K}$ is valid;
2. the main configuration $(\ell_0, \mathcal{K}, \texttt{false})$ leads to acceptance *iff* $\mathcal{K}$ is *not* valid.

Proof of Claim 3. We associate to $\mathcal{K}$ a natural number $\|\mathcal{K}\|$ defined as follows. Fix an ordering $\psi_1, \ldots, \psi_k$ of the formulas in $\mathsf{SD}(\varphi)$ such that for all $i \neq j$, $|\psi_i| > |\psi_j|$ implies $i < j$. First, we associate to $\mathcal{K}$ a $(k+1)$-tuple $\langle n_0, n_1, \ldots, n_k\rangle$ of natural numbers defined as follows: the first component n_0 in the tuple is the alternation depth $\Upsilon_w(\mathcal{K})$ and for the other components n_i with $1 \leq i \leq k$, n_i is the number of elements of $\mathcal{K}$ associated with the formula ψ_i (i.e., the number of elements of the form $(\psi_i, \langle T, V\rangle)$). Then, $\|\mathcal{K}\|$ is the position of the tuple $\langle n_0, n_1, \ldots, n_k\rangle$ along the total lexicographic ordering over $\mathbb{N}^{k+1}$. Note that if $\mathcal{K}$ is non-empty and universal, then since $\Upsilon_w(\widetilde{\mathcal{K}}) < \Upsilon_w(\mathcal{K})$, it holds that $\|\widetilde{\mathcal{K}}\| < \|\mathcal{K}\|$. Moreover, note that $\|\mathcal{K}\|$ strictly decreases at each iteration of the while loop in the procedures *checkTrue* and *checkFalse* (this is because at each iteration, $\Upsilon_w(\mathcal{K})$ does not increase, and an element of $\mathcal{K}$ is replaced with elements associated with smallest formulas).

The proof of Claim 3 is given by induction on $\|\mathcal{K}\|$. For the base case, $\|\mathcal{K}\| = 0$, hence $\mathcal{K}$ is empty and evidently valid. By construction, the procedure *checkTrue* accepts the empty set, while the procedure *checkFalse* rejects the empty set. Hence, for the base case, the result holds. For the induction step, let $\|\mathcal{K}\| > 0$, hence $\mathcal{K}$ is not empty. First, assume that $\mathcal{K}$ is universal. Recall that $\|\widetilde{\mathcal{K}}\| < \|\mathcal{K}\|$. Then:

- Property 1: $\mathcal{K}$ is valid $\iff$ (by the semantics of RML) for each $(\psi, \langle T, V\rangle) \in \widetilde{\mathcal{K}}$, $\{(\psi, \langle T, V\rangle)\}$ is not valid $\iff$ (by the induction hypothesis) for each $(\psi, \langle T, V\rangle) \in \mathcal{K}$, the main configuration $(\ell_0, \{(\psi, \langle T, V\rangle)\}, \texttt{false})$ leads to acceptance $\iff$ (by construction of the algorithm and since $\mathcal{K}$ is universal) the main configuration $(\ell_0, \mathcal{K}, \texttt{true})$ leads to acceptance.
- Property 2: $\mathcal{K}$ is *not* valid $\iff$ (by the semantics of RML) for some $(\psi, \langle T, V\rangle) \in \widetilde{\mathcal{K}}$, $\{(\psi, \langle T, V\rangle)\}$ is valid $\iff$ (by the induction hypothesis) for some $(\psi, \langle T, V\rangle) \in \widetilde{\mathcal{K}}$, the main configuration $(\ell_0, \{(\psi, \langle T, V\rangle)\}, \texttt{true})$ leads to acceptance $\iff$ (by construction of the algorithm and since $\mathcal{K}$ is universal) the main configuration $(\ell_0, \mathcal{K}, \texttt{false})$ leads to acceptance.

Hence, Properties 1 and 2 of Claim 3 hold if $\mathcal{K}$ is universal. Now, assume that the non-empty set $\mathcal{K}$ is not universal. We consider Property 2 of Claim 3 (the proof of Property 1 being dual). Let $(\psi, \langle T, V\rangle) \in \mathcal{K}$ be the pair selected by the procedure *checkFalse* in the iteration of the while loop associated with the main configuration $(\ell_0, \mathcal{K}, \texttt{false})$. Here, we examine the cases where $\psi = \Box\psi'$ or $\psi = \exists_r\psi'$ (the other cases being similar or simpler).

- Case $\psi = \Box\psi'$. Let $\mathcal{K}' = (\mathcal{K} \setminus \{(\psi, \langle T, V\rangle)\}) \cup \{(\psi', \langle T, V\rangle_{x_1}), \ldots, (\psi', \langle T, V\rangle_{x_k})\}$, where $x_1, \ldots, x_k$ are the children of the root of T. Note that $\|\mathcal{K}'\| < \|\mathcal{K}\|$. Then, we have that $\mathcal{K}$ is not valid $\iff$ (by the semantics of RML) $\mathcal{K}'$ is not valid $\iff$ (by the induction hypothesis) the main configuration $(\ell_0, \mathcal{K}', \texttt{false})$ leads to acceptance $\iff$ (by construction of the procedure *checkFalse*) the main configuration $(\ell_0, \mathcal{K}, \texttt{false})$ leads to acceptance.

- Case $\psi = \exists_r \psi'$. For this case, we crucially apply Theorem 2. First we show that Theorem 2 implies the following equivalence (*): $\langle T, V\rangle$ satisfies $\psi = \exists_r \psi' \iff$ there exists a certificate $\langle T', V'\rangle$ of ψ' satisfying ψ' which is a refinement of $\langle T, V\rangle$. The right-to-left implication is obvious. For the left-to-right implication, assume that there is a refinement of $\langle T, V\rangle$ which satisfies ψ'. By Remark 1, we can assume that such a refinement is a tree structure, say $\langle T'', V''\rangle$. By Theorem 2, there is a certificate $\langle T', V'\rangle$ of ψ' satisfying ψ' which is a refinement of $\langle T'', V''\rangle$. Thus, since the refinement relation is transitive, Condition (*) follows. Then, Property 2 of Claim 3 directly follows from the following chain of equivalences: $\mathcal{K}$ is not valid $\iff$ (by the semantics of RML) either $\mathcal{K} \setminus \{(\psi, \langle T, V\rangle)\}$ is not valid, or each refinement $\langle T', V'\rangle$ of $\langle T, V\rangle$ does not satisfy $\psi' \iff$ either $\mathcal{K} \setminus \{(\psi, \langle T, V\rangle)\}$ is not valid, or (by Condition (*)) each *certificate* $\langle T', V'\rangle$ of ψ' which is a refinement of $\langle T, V\rangle$ does not satisfy $\psi' \iff$ for each *certificate* $\langle T', V'\rangle$ of ψ' which is a refinement of $\langle T', V'\rangle$, $(\mathcal{K} \setminus \{(\psi, \langle T, V\rangle)\}) \cup \{(\psi', \langle T', V'\rangle)\}$ is not valid $\iff$ (by the induction hypothesis) for each *certificate* $\langle T', V'\rangle$ of ψ' which is a refinement of $\langle T, V\rangle$, the main configuration $(\ell_0, (\mathcal{K} \setminus \{(\psi, \langle T, V\rangle)\}) \cup \{(\psi', \langle T', V'\rangle)\}, \texttt{false})$ leads to acceptance $\iff$ (by construction of the procedure *checkFalse*) the main configuration $(\ell_0, \mathcal{K}, \texttt{false})$ leads to acceptance.

□

This concludes the proof of Theorem 12. □

By Theorem 12, the upper bounds illustrated in Figure 1 directly follow.

Corollary 13. *$\mathsf{SAT}(\mathsf{RML}) \in \mathsf{AEXP}_{\mathsf{pol}}$ and $\mathsf{SAT}(\mathsf{RML}^k) \in \Sigma_k^{\mathsf{EXP}}$ for each $k \geq 1$.*

5. 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* (TM, for short) 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 $\delta : (Q \setminus \{q_{acc}, q_{rej}\}) \times A \mapsto (Q \times A \times \{\leftarrow, \rightarrow\}) \cup \{1, \ldots, k\}$ 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, one of the following occurs:

- $\delta(q, a) \in Q \times A \times \{\leftarrow, \rightarrow\}$ (*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 ($\leftarrow$) or right ($\rightarrow$) in accordance with $\delta(q, a)$.[5]

[5] If the read/write head points to the left-most cell of the lth tape and $\delta(q, a)$ are of the form $(q', a, \leftarrow)$, then $\mathcal{M}$ moves to the rejecting state.

- $\delta(q,a) = h \in \{1,\ldots,k\}$ (*jump moves*): the read/write head jumps to the left-most cell of the hth tape and the state remains unchanged.

Initially, each tape contains a word in I^* and the read/write head points to the left-most cell of the first tape. Thus, an input of $\mathcal{M}$, called k-ary input, can be described by a tuple $(w_1,\ldots,w_k) \in (I^*)^k$, where for all $1 \leq i \leq k$, w_i represent the initial content of the ith tape. $\mathcal{M}$ *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 $(w_1,\ldots,w_k)$ is accepting. We consider the following problem.

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. Thus, even if $\mathcal{M}$ is polynomial-time bounded, it operates on an input (witness) whose size is exponential in 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 14. *The Linear Alternation Problem is $\mathsf{AEXP}_{\mathsf{pol}}$-complete and for all $k \geq 1$, the k-Alternation Problem is Σ_k^{EXP}-complete.*

The proof of Proposition 14 is standard. However, for completeness, we give a proof for the hardness results in Proposition 14, which are the relevant ones in this context. Then, in Subsection 5.1, we describe polynomial-time reductions from the Linear Alternation Problem to $\mathsf{SAT}(\mathsf{RML})$, and from the k-Alternation Problem to $\mathsf{SAT}(\mathsf{RML}^{k+1})$ (for each $k \geq 1$).

The lower bounds in Proposition 14 directly follows from the following two lemmata.

Lemma 15. *Let $k \geq 1$ and $\mathcal{M}_\mathcal{A}$ be a singly exponential-time bounded ATM with at most $k-1$ alternations and such that the initial state is existential. Moreover, let $c \geq 1$ be an integer constant such that for each input α, when started on α, $\mathcal{M}_\mathcal{A}$ reaches a terminal configuration in at most $2^{|\alpha|^c}$ steps. Then, given an input α, one can construct in time polynomial in α and the size of $\mathcal{M}_\mathcal{A}$ an instance $(k, 2|\alpha|^c, \mathcal{M})$ of the* Alternation Problem *such that the instance is positive* iff *$\mathcal{M}_\mathcal{A}$ accepts α.*

Lemma 16. *Let $\mathcal{M}_\mathcal{A}$ be a singly exponential-time bounded ATM with a polynomial-time bounded number of alternations. Moreover, let $c \geq 1$ and $c_a \geq 1$ be integer constants such that for each input α, when started on α, $\mathcal{M}_\mathcal{A}$ has at most $|\alpha|^{c_a}$ alternations and $\mathcal{M}_\mathcal{A}$ reaches a terminal configuration in at most $2^{|\alpha|^c}$ steps. Then, given an input α, one can construct in time polynomial in α and the size of $\mathcal{M}_\mathcal{A}$ an instance $(2|\alpha|^{max\{c,c_a\}}, 2|\alpha|^{max\{c,c_a\}}, \mathcal{M})$ of the* Alternation Problem *such that the instance is positive* iff *$\mathcal{M}_\mathcal{A}$ accepts α.*

The proofs of Lemmata 15 and 16 are very similar, so that we prove only Lemma 15.

Proof of Lemma 15. Let $\mathcal{M}_\mathcal{A}$, c, and k as in the statement of Lemma 15. Let $I_\mathcal{A}$ (resp., $A_\mathcal{A}$) be the input (resp., work) alphabet of $\mathcal{M}_\mathcal{A}$, where $I_\mathcal{A} \subset A_\mathcal{A}$, and Q be the set of $\mathcal{M}_\mathcal{A}$-states. Fix an input $\alpha \in I_\mathcal{A}^*$. An α-configuration is a word in $A_\mathcal{A}^* \cdot (Q \times A_\mathcal{A}) \cdot A_\mathcal{A}^*$ of length exactly $2^{|\alpha|^c}$. Note that any configuration of $\mathcal{M}_\mathcal{A}$ reachable from the input α can be encoded by an α-configuration. We denote by C_α the initial (existential) α-configuration associated with the input α. A *partial computation* of $\mathcal{M}_\mathcal{A}$ is a finite sequence $\pi = C_1, \ldots, C_p$ of α-configurations such that $p \leq 2^{|\alpha|^c}$ and for each $1 \leq i < p$, C_{i+1} is a $\mathcal{M}_\mathcal{A}$-successor of C_i (note that a computation of $\mathcal{M}_\mathcal{A}$ over α is a partial computation). We say that π is *uniform* if additionally one of the following holds:

- C_p is terminal and π visits only existential n-configurations;
- C_p is terminal and π visits only universal n-configurations;
- $p > 1$, C_p is existential and for each $1 \leq h < p$, C_h is universal;
- $p > 1$, C_p is universal and for each $1 \leq h < p$, C_h is existential.

Let $\diamondsuit$ be a fresh symbol and $I = A_\mathcal{A} \cup \{\diamondsuit\}$. The *code* of a partial computation $\pi = C_1, \ldots, C_p$ is the word over I of length exactly $2^{2|\alpha|^c}$ given by $C_1, \ldots, C_p, C^0_{p+1}, \ldots, C^0_{2^{|\alpha|^c}}$, where $C^0_h \in \{\diamondsuit\}^{2^{|\alpha|^c}}$ for all $p + 1 \leq h \leq 2^{|\alpha|^c}$. We construct a polynomial-time bounded k-ary deterministic Turing Machine $\mathcal{M}$, which satisfies Lemma 15 for the given input α of $\mathcal{M}_\mathcal{A}$, as follows. The input alphabet of $\mathcal{M}$ is I. Given a k-ary input $(w_1, \ldots, w_k) \in (I^*)^k$, $\mathcal{M}$ operates in k-steps. At step i $(1 \leq i \leq k)$, the behavior of $\mathcal{M}$ is as follows, where $n = 2|\alpha|^c$ and for a partial computation $\pi = C_1, \ldots, C_p$, $\mathsf{first}(\pi) = C_1$ and $\mathsf{last}(\pi) = C_p$:

- *First step:* $i = 1$.
    1. If $w_1 \in I^{2^n}$ and w_1 encodes a uniform partial computation π_1 of $\mathcal{M}_\mathcal{A}$ from C_α, then the behavior is as follows. If $\mathsf{last}(\pi_1)$ is accepting (resp., rejecting), then $\mathcal{M}$ accepts (resp., rejects) the input . If instead $\mathsf{last}(\pi_1)$ is not a terminal configuration, then $\mathcal{M}$ goes to step $i + 1$.
    2. Otherwise, $\mathcal{M}$ *rejects* the input.
- $i > 1$.
    1. If $w_i \in I^{2^n}$ and w_i encodes a uniform partial computation π_i of $\mathcal{M}_\mathcal{A}$ such that $\mathsf{first}(\pi_i) = \mathsf{last}(\pi_{i-1})$, where π_{i-1} is the uniform partial computation encoded by w_{i-1}, then the behavior is as follows. If $\mathsf{last}(\pi_i)$ is accepting (resp., rejecting), then $\mathcal{M}$ accepts (resp., rejects) the input. If instead $\mathsf{last}(\pi_i)$ is not a terminal configuration, then $\mathcal{M}$ goes to step $i + 1$.
    2. Otherwise, if i is odd (resp., even), then $\mathcal{M}$ *rejects* (resp., *accepts*) the input.

Note that Conditions 1 in the steps above can be checked by $\mathcal{M}$ in polynomial time (in the size of the input) by using the transition function of $\mathcal{M}_\mathcal{A}$ and n-bit counters. Hence, $\mathcal{M}$ is a polynomial-time bounded k-ary deterministic Turing Machine which, evidently, can be constructed in time polynomial in n and the size of $\mathcal{M}_\mathcal{A}$. Now, we prove that the construction is correct, i.e. $(k, n, \mathcal{M})$ is a positive instance of the Alternation Problem iff $\mathcal{M}_\mathcal{A}$ accepts α. For each $1 \leq \ell \leq k$, let $\mathsf{Q}_\ell = \exists$ if ℓ is odd, and $\mathsf{Q}_\ell = \forall$ otherwise. Since C_α is existential, $\mathcal{M}_\mathcal{A}$ accepts α iff there is a uniform

partial computation π_1 of $\mathcal{M}_\mathcal{A}$ from C_α such that $\mathsf{last}(\pi_1)$ leads to acceptance. Moreover, for each $w_1 \in I^{2^n}$, $\mathcal{M}$ accepts an input of the form $(w_1, w'_2, \ldots, w'_k)$ *only if* w_1 encodes a non-rejecting uniform partial computation of $\mathcal{M}_\mathcal{A}$ from C_α. Thus, since $\mathsf{Q}_1 = \exists$, correctness of the construction directly follows from the following claim.

Claim. Let $1 \leq \ell \leq k$ and $\pi = \pi_1 \ldots \pi_\ell$ be a partial computation of $\mathcal{M}_\mathcal{A}$ from C_α such that π_ℓ is uniform and for each $1 \leq h < \ell$, π_h is non-empty and $\pi_h \cdot \mathsf{first}(\pi_{h+1})$ is uniform as well. Let $w_\ell \in I^{2^n}$ be the word encoding π_ℓ and for each $1 \leq h < l$, $w_h \in I^{2^n}$ be the word encoding $\pi_h \cdot \mathsf{first}(\pi_{h+1})$. Then, $\mathsf{last}(\pi_\ell)$ leads to acceptance if and only if

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

Proof of the claim. The proof is by induction on $k - \ell$.

Base Step: $\ell = k$. Note that in this case $\mathsf{last}(\pi_k)$ is a terminal configuration (otherwise, the number of alternations of existential and universal configurations along π would be greater than $k - 1$). Thus, we need to show that $\mathsf{last}(\pi_k)$ is accepting iff $\mathcal{M}(w_1, \ldots, w_k)$. By construction, when started on the input $(w_1, \ldots, w_k)$, $\mathcal{M}$ reaches the kth step and Condition 1 in this step is satisfied. Moreover, either $\mathsf{last}(\pi_k)$ is accepting and $\mathcal{M}$ accepts the input $(w_1, \ldots, w_k)$, or $\mathsf{last}(\pi_k)$ is rejecting and $\mathcal{M}$ rejects the input $(w_1, \ldots, w_k)$. Hence, the result follows.

Induction Step: $\ell < k$. First, assume that $\mathsf{last}(\pi_\ell)$ is a terminal configuration. By construction on any input of the form $(w_1, \ldots, w_\ell, w'_{\ell+1}, \ldots, w'_k)$, $\mathcal{M}$ reaches the ℓth step and Condition 1 in this step is satisfied. Moreover, either $\mathsf{last}(\pi_\ell)$ is accepting and $\mathcal{M}$ accepts the input $(w_1, \ldots, w_\ell, w'_{\ell+1}, \ldots, w'_k)$, or $\mathsf{last}(\pi_\ell)$ is rejecting and $\mathcal{M}$ rejects the input $(w_1, \ldots, w_\ell, w'_{\ell+1}, \ldots, w'_k)$. Hence, in this case the result holds. Now, assume that $\mathsf{last}(\pi_\ell)$ is not terminal. We consider the case where $\ell + 1$ is even (the other case being similar). Then, $\mathsf{Q}_{\ell+1} = \forall$. Since C_α is existential and $\mathsf{last}(\pi_\ell)$ is not terminal, by hypothesis, $\mathsf{last}(\pi_\ell)$ must be an universal configuration. First, assume that $\mathsf{last}(\pi_\ell)$ leads to acceptance. Let $w_{\ell+1} \in I^{2^n}$. By construction on any input of the form $(w_1, \ldots, w_\ell, w_{\ell+1}, w'_{\ell+2} \ldots, w'_k)$, $\mathcal{M}$ reaches the $(\ell + 1)$th step. If $w_{\ell+1}$ satisfies Condition 2 in this step, then since $\ell + 1$ is even, $\mathcal{M}$ accepts the input. Hence, $\mathsf{Q}_{\ell+2} x_{\ell+2} \in I^{2^n}. \ldots \mathsf{Q}_k x_k \in I^{2^n}. \mathcal{M}(w_1, \ldots, w_\ell, w_{\ell+1}, x_{\ell+2}, \ldots, x_k)$. Otherwise, $w_{\ell+1}$ encodes a uniform partial computation $\pi_{\ell+1}$ of $\mathcal{M}_\mathcal{A}$ from $\mathsf{last}(\pi_\ell)$. Since $\mathsf{last}(\pi_\ell)$ leads to acceptance and $\mathsf{last}(\pi_\ell)$ is universal, $\mathsf{last}(\pi_{\ell+1})$ leads to acceptance as well. Thus, applying the induction hypothesis to the partial computation $\pi_1 \ldots \pi_{\ell-1} \pi'_\ell \pi_{\ell+1}$ (where π'_ℓ is obtained from π_ℓ by removing $\mathsf{last}(\pi_\ell)$), it follows that $\mathsf{Q}_{\ell+2} x_{\ell+2} \in I^{2^n}. \ldots \mathsf{Q}_k x_k \in I^{2^n}. \mathcal{M}(w_1, \ldots, w_\ell, w_{\ell+1}, x_{\ell+2}, \ldots, x_k)$. Thus, the previous condition holds for each $w_{\ell+1} \in I^{2^n}$. Since $\mathsf{Q}_{\ell+1} = \forall$, it follows that Condition (1) holds. For the converse direction, assume that Condition (1) holds. Let $\pi_{\ell+1}$ be any uniform partial computation of $\mathcal{M}_\mathcal{A}$ from $\mathsf{last}(\pi_\ell)$. We need to show that $\mathsf{last}(\pi_{\ell+1})$ leads to acceptance. Since Condition (1) holds and $\mathsf{Q}_{\ell+1} = \forall$, we can apply the induction hypothesis to the partial computation $\pi_1 \ldots \pi_{\ell-1} \pi'_\ell \pi_{\ell+1}$ (where π'_ℓ is obtained from π_ℓ by removing $\mathsf{last}(\pi_\ell)$). Hence, the result follows. □

This concludes the proof of Lemma 15.

5.1. Reductions from the Alternation Problem

In this Subsection, we show that the Linear Alternation Problem reduces in polynomial time to $\mathsf{SAT}(\mathsf{RML})$, and for each $k \geq 1$, the k-Alternation Problem reduces in polynomial time to $\mathsf{SAT}(\mathsf{RML}^{k+1})$.

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 $1 \leq i \leq k$. Hence, on a (k, n)-input, $\mathcal{M}$ reaches a terminal configuration in at most $2^{\mathsf{c}(k,n)}$ steps, where $\mathsf{c}(k, n) := c\cdot(n+\lceil\log k\rceil)$. Now, we observe that for the Linear Alternation Problem and the k-Alternation Problem, $\mathsf{c}(k, n)$ is a function of just n (rather than n and k). In particular, for both the problems, we can construct an integer constant e such that $\mathsf{c}(k, n) \leq e \cdot n$ as follows: for the Linear Alternation Problem, we set $e = 2c$, and for the k-Alternation Problem, we set $e = c(k + 1)$. Hence, for the considered problems, $2^{e\cdot n}$ represents an upper bound on the number of computational steps of $\mathcal{M}$ from (k, n)-inputs. Moreover, without loss of generality, we can make the following assumption: on a (k, n)-input, $\mathcal{M}$ reaches a terminal configuration in exactly $2^{e\cdot n}$ steps. In the following, in order to simplify the presentation, we additionally assume that $e = 1$. Note that our results continue to hold if $e \neq 1$ (in the construction of the RML formulas given in the rest of this section, it suffices to replace n with $e \cdot n$).

Thus, 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^n such that for some $1 \leq \ell \leq k$, C_ℓ is of the form $w \cdot (q, a) \cdot w' \in A^* \cdot (Q \times A) \cdot A^*$ (where $w \cdot a \cdot w'$ denotes the ℓth tape content, q the current state, and the read/write head scans the $(|w| + 1)$th cell of the ℓth tape), and for $i \neq \ell$, $C_i \in A^{2^n}$ (denoting that the tape content of the ith tape is C_i). For a (k, n)-input $(a \cdot w_1, \ldots, w_k)$ with $a \in I$, the associated *initial* (k, n)-*configuration* is $((q_0, a) \cdot w_1, \ldots, w_k)$ (i.e., initially, the read/write head scans the first cell of the first tape). Moreover, the computations of $\mathcal{M}$ from (k, n)-inputs, called (k, n)-*computations*, can be described by sequences π of exactly 2^n (k, n)-configurations. In the rest of this section, we prove the following result.

Theorem 17. *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 an* $\mathsf{RML}^{\forall}$ *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 14 and Theorem 17, we obtain the following.

Corollary 18. $\mathsf{SAT}(\mathsf{RML})$ *is* AEXP_{pol}*-hard,* $\mathsf{SAT}(\mathsf{RML}^{\forall})$ *is* $\mathsf{NEXPTIME}$*-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 17, 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_+) such that $u \in A\cup(Q\times A)$ and $u_-, u_+ \in A\cup(Q\times A)\cup\{\bot\}$ ($\bot$ is for undefined). Intuitively, u_-, u, and u_+ represent three consecutive symbols in some component C_ℓ of a (k, n)-configuration $\overrightarrow{C} = (C_1, \ldots, C_k)$, where $u_- = \bot$ (resp., $u_+ = \bot$) iff u is the first (resp., the last) symbol of C_ℓ. In the encoding of a (k, n)-computation $\pi = \overrightarrow{C}_0, \ldots, \overrightarrow{C}_{2^n-1}$, for each (k, n)-configuration $\overrightarrow{C}_i$ along π, we keep track of the position i of $\overrightarrow{C}_i$ along π; in particular, since $0 \leq i \leq 2^n - 1$, i can be encoded by a sequence of n bits. Moreover, in the encoding of a

(k,n)-configuration $\overrightarrow{C} = (C_1, \ldots, C_k)$, for each component $C_\ell = C_\ell(0) \ldots C_\ell(2^n - 1)$ of $\overrightarrow{C}$, each symbol $C_\ell(j)$ of C_ℓ is encoded by a word which keeps track of ℓ (by the symbol $arg_\ell \in P$), of the triple $(C_\ell(j-1), C_\ell(j), C_\ell(j+1))$, and of the binary encoding of position j (a sequence of n bits). This leads to the following definition.

An *extended TM block ext_bl* is a word over 2^P of length $2n+2$ of the form

$$ext\_bl = \{bit_1\} \cdot \ldots \cdot \{bit_n\} \cdot bl \quad \text{with } bl = \{bit'_1\} \cdot \ldots \cdot \{bit'_n\} \cdot \{arg_\ell\} \cdot \{t\}$$

where $1 \leq \ell \leq k$ and $t \in \Lambda$. We say that *bl* is a *TM block*, t is the *content* of *ext_bl* and *bl*, and ℓ is the *component number* of *ext_bl* and *bl*. Moreover, the *time position number* of *ext_bl* is the integer $\underline{i} \in [0, 2^n - 1]$ whose binary code is $bit_1, \ldots, bit_n$, while the *tape position number* of *ext_bl*, also called *the position number of bl*, is the integer $\underline{j} \in [0, 2^n - 1]$ whose binary code is $bit'_1, \ldots, bit'_n$. Intuitively, *ext_bl* encodes the triple $t = (C_\ell(\underline{j} - 1), C_\ell(\underline{j}), C_\ell(j+1))$ (where $C_\ell(j-1) = \perp$ if $j = 0$, and $C_\ell(j+1) = \perp$ if $j = 2^n - 1$) of the ℓth component C_ℓ associated with the $\underline{i}$th (k,n)-configuration of some (k,n)-computation. Thus, the time position number keeps tracks of the position of a (k,n)-configuration along a (k,n)-computation. Moreover, for a fixed time position number, the component number is used to specify a component of the given (k,n)-configuration, and the tape position number refers to the position of a specific symbol in the given component.

For an arbitrary sequence $\pi = \overrightarrow{C}_0, \ldots, \overrightarrow{C}_{2^n-1}$ of 2^n (k,n)-configurations,[6] we can encode π by the set $S_{ext\_bl}(\pi)$ of extended blocks defined as follows: *ext_bl* $\in S_{ext\_bl}(\pi)$ *if and only if* there are $0 \leq i, j \leq 2^n - 1$ and $0 \leq \ell \leq k$ such that *ext_bl* has time position number i, tape position number j, component number ℓ, and content $(C(j-1), C(\underline{j}), C(j+1))$, where C is the ℓth component of $\overrightarrow{C}_i$ (with $C(j-1) = \perp$ if $j = 0$, and $C(j+1) = \perp$ if $j = 2^n - 1$). The tree representation of the set $S_{ext\_bl}(\pi)$ is defined as follows.

Definition 8 ((k,n)-computation tree codes). *A* (k,n)-computation tree code is *a tree structure* $\langle T, V \rangle$ *over* P *satisfying the following:*

1. *Each path of* $\langle T, V \rangle$ *from the root has length* $2n+2$*, and each node is labeled by exactly one proposition in* P *(i.e.,* $V(x)$ *is a singleton for each* $p \in P$*). Moreover,* $\langle T, V \rangle$ *satisfies the* ML*-formula*

$$\bigwedge_{i=1}^{2n} \Box^{i-1}((\Diamond 0 \wedge \Diamond 1) \wedge \Box(0 \vee 1)) \wedge \Box^{2n}(\bigwedge_{\ell=1}^{k} \Diamond arg_\ell \wedge \Box \bigvee_{\ell=1}^{k} arg_\ell) \wedge \Box^{2n+2} \bigvee_{t \in \Lambda} t$$

   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* $\pi = \overrightarrow{C}_0, \ldots, \overrightarrow{C}_{2^n-1}$ *of* 2^n (k,n)*-configurations such that the set of extended TM blocks of* $\langle T, V \rangle$ *corresponds to the set* $S_{ext\_bl}(\pi)$ *(note that* π *is uniquely determined). For each* $0 \leq i \leq 2^n - 1$*, we say that* $\overrightarrow{C}_i$ *is the* (k,n)*-configuration with time position number* i *encoded by* $\langle T, V \rangle$*.*

[6] Note that we do not require that π is a (k,n)-computation.

A (k,n)-computation tree code $\langle T,V\rangle$ is initialized *if the (k,n)-configuration with time position number 0 encoded by $\langle T,V\rangle$ is initial.*[7]

Note that Property 1 of Definition 8 (independently of Property 2) ensures the following: for all $0 \leq i,j \leq 2^n-1$ and $1 \leq \ell \leq k$, there is a path encoding an extended block *ext_bl* with time position number i, component number ℓ, and tape position number j. Moreover, note that an initialized (k,n)-computation tree code encodes a sequence of 2^n (k,n)-configurations which starts at an initial (k,n)-configuration.

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 9, where for an input $w = a_0 \dots a_{2^n-1} \in I^{2^n}$ and an extended block *ext_bl* with tape position number j and component number h, we say that *ext_bl* is *initial-consistent* with w if the content of *ext_bl* satisfies the following:

$$\text{content of } \textit{ext\_bl} \;=\; \begin{cases} (\bot,(q_0,a_0),a_1) & \text{if } h=1 \text{ and } j=0 \\ ((q_0,a_0),a_1,a_2) & \text{if } h=1 \text{ and } j=1 \\ (a_{j-1},a_j,a_{j+1}) & \text{otherwise} \end{cases}$$

where $a_{-1} = \bot$ and $a_{2^n} = \bot$. Intuitively, the TM block of *ext_bl* encodes the jth symbol of the hth component of any initial (k,n)-configuration associated with a (k,n)-input whose hth component is w.

Definition 9 (Initialized full (k,n)-tree codes). *Let $1 \leq \ell \leq k$. An* ℓ-initialized full (k,n)-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 8. Moreover, let $\nu = z_0,\dots,z_{2n+1},z_{2n+2}$ be a path of $\langle T,V\rangle$ (from the root) encoding an extended TM block ext_bl with component number h and time position number i such that either $h > \ell$ or $i > 0$. Then, for each $t \in \Lambda$, there is a child z of z_{2n+1} which is labeled by $\{t\}$.*

2. ℓ-initialization requirement. *There are $w_1,\dots,w_\ell \in I^{2^n}$ such that for all $1 \leq h \leq \ell$, $0 \leq j \leq 2^n-1$, and extended blocks ext_bl of $\langle T,V\rangle$ with component number h, time position number 0, and tape position number j,*[8] *ext_bl is initial-consistent with the input w_h. We say that $w_1,\dots,w_\ell \in I^{2^n}$ is the ℓ-ary input (which is uniquely determined) associated with $\langle T,V\rangle$ and we write $\langle T,V\rangle(w_1,\dots,w_\ell)$.*

Note that Property 1 of Definition 9 ensures that for each extended TM block *ext_bl* with component number h and time position number i such that either $h > \ell$ or $i > 0$, there is a path of $\langle T,V\rangle$ encoding *ext_bl*. Intuitively, an ℓ-initialized full (k,n)-tree code $\langle T,V\rangle$ associated with an ℓ-ary input $w_1,\dots,w_\ell \in I^{2^n}$ encodes all the possible (k,n)-computations from (k,n)-inputs of the form $(w_1,\dots,w_\ell,w'_{\ell+1},\dots,w'_k)$ for arbitrary words $w'_{\ell+1},\dots,w'_k \in I^{2^n}$. More precisely, by construction, the following holds.

Proposition 19. *Let $1 \leq \ell \leq k-1$, $w_1,\dots,w_\ell \in I^{2^n}$, and $\langle T,V\rangle$ be an ℓ-initialized full (k,n)-tree code such that $\langle T,V\rangle(w_1,\dots,w_\ell)$. Then:*

[7] Recall that an initial (k,n)-configuration $\overrightarrow{C}$ is the (k,n)-configuration initially assumed by $\mathcal{M}$ when started on a (k,n)-input, i.e., $\overrightarrow{C}$ is of the form $((q_0,a)\cdot w_1,\dots,w_k)$ for some (k,n)-input $(a\cdot w_1,\dots,w_k)$.

[8] Property 1 ensures that such *ext_bl* exists.

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

2. *Assume that* $\ell = k-1$*. Then, for each* $w \in I^{2^n}$*, there is a refinement of* $\langle T, V\rangle$ *which is a* (k,n)*-computation tree code encoding the* (k,n)*-computation from the* (k,n)*-input* $(w_1, \ldots, w_{k-1}, w)$*. Moreover, for each refinement* $\langle T_r, V_r\rangle$ *of* $\langle T, V\rangle$ *which is a* (k,n)*-computation tree code encoding a* (k,n)*-computation* π*, there is* $w \in I^{2^n}$ *such that* π *is the* (k,n)*-computation from the* (k,n)*-input* $(w_1, \ldots, w_{k-1}, w)$*.*

The core result in the proposed reduction is represented by the following lemma, where a 0-initialized full (k,n)-tree code is an arbitrary tree structure.

Lemma 20. *One can build in time polynomial in* n*,* k*, and the size of the TM* $\mathcal{M}$*,*

1. *an* $\mathsf{RML}^\forall$ *formula* φ^ℓ_{init} *over* P *(for each* $1 \leq \ell \leq k$*) such that given a* refinement $\langle T_r, V_r\rangle$ *of an* $\ell-1$*-initialized full* (k,n)*-tree code,* $\langle T_r, V_r\rangle$ *satisfies* φ^ℓ_{init} if and only if $\langle T_r, V_r\rangle$ *is an* ℓ*-initialized full* (k,n)*-tree code;*

2. *an* $\mathsf{RML}^\forall$ *formula* $\varphi_{in\_tree\_code}$ *over* P *such that given a* refinement $\langle T_r, V_r\rangle$ *of a* $k-1$*-initialized full* (k,n)*-tree code,* $\langle T_r, V_r\rangle$ *satisfies* $\varphi_{in\_tree\_code}$ iff $\langle T_r, V_r\rangle$ *is an* initialized (k,n)*-computation tree code.*

3. *an* $\mathsf{RML}^\forall$ *formula* $\varphi_{faithful}$ *over* P *such that given a* (k,n)*-computation tree code* $\langle T, V\rangle$*,* $\langle T, V\rangle$ *satisfies* $\varphi_{faithful}$ iff *the sequence of* (k,n)*-configurations encoded by* $\langle T, V\rangle$ *is faithful to the evolution of* $\mathcal{M}$*.*

4. *an* 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 time position number* 2^n-1 *encoded by* $\langle T, V\rangle$ *is accepting in* $\mathcal{M}$*.*

Proof. Since Property 4 is trivial, we only prove Properties 1–3 of the lemma. We use the following ML formula $\Phi_{complete}$ characterizing the tree structures such that each path from the root has length $2n+2$.

$$\Phi_{complete} := \Box^{2n+3}\mathtt{false} \wedge \bigwedge_{i=0}^{2n+1} \Box^i \Diamond \mathtt{true}$$

Proof of Property 1. Let $1 \leq \ell \leq k$. We construct in time polynomial in n, k, and the size of the TM $\mathcal{M}$, an $\mathsf{RML}^\forall$ formula φ^ℓ_{init} over P such that given a *refinement* $\langle T_r, V_r\rangle$ of an $\ell-1$-initialized full (k,n)-tree code, $\langle T_r, V_r\rangle$ satisfies φ^ℓ_{init} *iff* $\langle T_r, V_r\rangle$ is an ℓ-initialized full (k,n)-tree code. Hence, Property 1 of Lemma 20 follows. We assume that $\ell > 1$. The case $\ell = 1$ is similar except we additionally need to impose that each node in the given refinement is labeled by exactly one atomic proposition in P: note that this requirement can be expressed by the ML formula $\bigwedge_{i=0}^{2n+2} \Box^i[\bigvee_{p\in P}(p \wedge \bigwedge_{q\in P\setminus\{p\}} \neg q)]$. The $\mathsf{RML}^\forall$ formula φ^ℓ_{init} is defined as follows:

$$\varphi^\ell_{init} := \varphi^\ell_{full} \wedge \varphi^\ell_{unique} \wedge \varphi^\ell_{conf}$$

where for a given refinement $\langle T_r, V_r\rangle$ of an $\ell-1$-initialized full (k,n)-tree code, φ^{ℓ}_{full} is an ML formula ensuring that $\langle T_r, V_r\rangle$ satisfies the fullness requirement in Definition 9, while φ^{ℓ}_{unique} and φ^{ℓ}_{conf} are $\mathsf{RML}^{\forall}$ formulas ensuring that $\langle T_r, V_r\rangle$ satisfies the ℓ-initialization requirement in Definition 9. Note that since $\langle T_r, V_r\rangle$ is a refinement of an $\ell-1$-initialized full (k,n)-tree code, the fullness requirement (enforced by formula φ^{ℓ}_{full}) ensures the fulfillment of the $\ell-1$-initialization requirement too. Hence, in order to ensure also the ℓ-initialization requirement, we just need to impose that the set of extended blocks of time position number 0 and component number ℓ encodes the ℓth component of some initial (k,n)-configuration. This is achieved by the following two requirements:

1. *Uniqueness initial condition*: for all extended TM blocks *ext_bl* and *ext_bl'* having component number ℓ, time position number 0, and the *same* tape position number, the following holds: *ext_bl* and *ext_bl'* have the *same* content which is of the form (u_-, a, u_+) for some $a \in I$.
2. *Configuration well-formedness initial condition*: For all extended TM blocks *ext_bl* and *ext_bl'* having component number ℓ, time position number 0, and such that the associated TM blocks *bl* and *bl'* have position number j and $j+1$ for some $0 \leq j < 2^n-1$ (consecutive ℓ-TM blocks), the following holds: the contents of *ext_bl* and *ext_bl'* are of the form (u_-, u, u_+) and (u, u_+, u_{++}), respectively. Additionally, if $j = 0$ (resp., $j+1 = 2^n-1$), then the content of *ext_bl* (resp., *ext_bl'*) is of the form $(\bot, u, u_+)$ (resp., $(u_-, u, \bot)$).

The $\mathsf{RML}^{\forall}$ formula φ^{ℓ}_{unique} enforces the first requirement, while the $\mathsf{RML}^{\forall}$ formula φ^{ℓ}_{conf} enforces the second one. Formally, the formulas φ^{ℓ}_{full}, φ^{ℓ}_{unique}, and φ^{ℓ}_{conf} are defined as follows.

$$\begin{aligned}\varphi^{\ell}_{full} \;:=\; & \Big(\bigwedge_{i=1}^{2n} \Box^{i-1}(\Diamond 0 \wedge \Diamond 1)\Big) \wedge \Big(\Box^{2n} \bigwedge_{i=1}^{k} \Diamond arg_i\Big) \wedge \Big(\Box^{2n+1} \bigwedge_{i=\ell+1}^{k} (arg_i \rightarrow \bigwedge_{t\in\Lambda} \Diamond t)\Big) \;\wedge \\ & \Big(\bigwedge_{i=1}^{n} \Box^{i}(1 \rightarrow \Box^{2n-i+1} \bigwedge_{t\in\Lambda} \Diamond t)\Big) \;\wedge\; \Phi_{complete}\end{aligned}$$

Note that since $\langle T_r, V_r\rangle$ is a refinement of an $\ell-1$-initialized full (k,n)-tree code and $\ell > 1$, each node of T_r is labeled by exactly one atomic proposition. Hence, it easily follows that φ^{ℓ}_{full} ensures the fullness requirement.

$$\begin{aligned}\varphi^{\ell}_{unique} \;:=\; \forall_r\Big(& \underbrace{\Big(\Phi_{complete} \wedge \bigwedge_{i=1}^{n} \Box^{i} 0 \wedge \neg \bigvee_{i=1}^{n} (\Diamond^{n+i} 0 \wedge \Diamond^{n+i} 1) \wedge \Box^{2n+1} arg_\ell\Big)}_{\text{select } \ell\text{-TM blocks with the same position number of extended TM blocks of time position number 0}} \\ & \longrightarrow \bigvee_{a\in I} \bigvee_{(u_-,a,u_+)\in\Lambda} \Box^{2n+2}(u_-, a, u_+)\Big)\end{aligned}$$

$$\varphi^{\ell}_{conf} \;:=\; \forall_r\Big(\underbrace{\Big(\Phi_{complete} \wedge \bigwedge_{i=1}^{n} \Box^{i} 0 \;\wedge\; \Box^{2n+1} arg_\ell \;\wedge\; \varphi_{inc}\Big)}_{\text{select two consecutive } \ell\text{-TM blocks of extended TM blocks of time position number 0}} \longrightarrow \varphi_{check}\Big)$$

where φ_{inc} and φ_{check} are ML formulas defined below. In particular, φ_{inc} allows to "select" only TM blocks having position numbers i and $i+1$ for some $0 \leq i < 2^n - 1$, while φ_{check} checks the consistency of the selected consecutive ℓ-TM blocks associated with extended TM blocks of time position number 0. Note that for two TM blocks $bl = \{bit_1\}, \ldots, \{bit_n\}, \ldots$ and $bl' = \{bit'_1\}, \ldots, \{bit'_n\}, \ldots$, their position numbers are consecutive iff there is $1 \leq i \leq n$ such that: (1) $bit_i = 0$ and $bit'_i = 1$, (2) for each $1 \leq j \leq i-1$, $bit_j = 1$ and $bit'_j = 0$, and (3) for each $i+1 \leq j \leq n$, $bit_j = bit'_j$.

$$\varphi_{inc} \;:=\; \Diamond^{n+1}0 \wedge \Diamond^{n+1}1 \wedge \bigvee_{i=1}^{n}\big(\varphi^i_{inc} \wedge \neg \bigvee_{j=i+1}^{n}(\Diamond^{n+j}0 \wedge \Diamond^{n+j}1)\big)$$

$$\varphi^i_{inc} := \begin{cases} \mathtt{true} & \text{if } i = 1 \\ \Box^{n+1}\big(0 \rightarrow (\Box^{i-1}1 \wedge \bigwedge_{j=1}^{i-2}\Box^j 0)\big) \wedge \Box^{n+1}\big(1 \rightarrow (\Box^{i-1}0 \wedge \bigwedge_{j=1}^{i-2}\Box^j 1)\big) & \text{otherwise} \end{cases}$$

$$\varphi_{check} \;:=\; \bigvee_{(u_-,u,u_+),(u,u_+,u_{++})\in\Lambda} \bigvee_{i=1}^{n}\Big((\neg \bigvee_{j=i+1}^{j=n}(\Diamond^{n+j}0 \wedge \Diamond^{n+j}1)) \wedge \Diamond^{n+i}0 \wedge \Diamond^{n+i}1 \wedge \underbrace{\Box^{n+i}(0 \rightarrow \Box^{n-i+2}(u_-,u,u_+)) \wedge \Box^{n+i}(1 \rightarrow \Box^{n-i+2}(u,u_+,u_{++}))}_{\text{content consistency for consecutive TM blocks in the same component of a } (k,n)\text{-configuration}}\Big)$$

$$\wedge$$

$$\Big(\Diamond^n(\underbrace{\Diamond(0 \wedge \Diamond(0 \wedge \ldots \wedge \Diamond 0)\ldots)}_{n \text{ times}}))) \longrightarrow \Diamond^n(\underbrace{\Diamond(0 \wedge \Diamond(0 \wedge \ldots \wedge \Diamond(0}_{n \text{ times}} \wedge \bigvee_{(\bot,u,u_+)\in\Lambda} \Diamond^2(\bot,u,u_+))\ldots))))\Big)$$

$$\wedge$$

$$\Big(\Diamond^n(\underbrace{\Diamond(1 \wedge \Diamond(1 \wedge \ldots \wedge \Diamond 1)\ldots)}_{n \text{ times}}))) \longrightarrow \Diamond^n(\underbrace{\Diamond(1 \wedge \Diamond(1 \wedge \ldots \wedge \Diamond(1}_{n \text{ times}} \wedge \bigvee_{(u_-,u,\bot)\in\Lambda} \Diamond^2(u_-,u,\bot))\ldots))))\Big)$$

Proof of Property 2. First, we observe that one can easily construct in time polynomial in n and k, an ML formula which is satisfied by a (k,n)-computation tree code $\langle T, V\rangle$ iff $\langle T, V\rangle$ is initialized. Thus, it suffices to show that one can construct in time polynomial in n, k, and the size of the TM $\mathcal{M}$, an $\mathsf{RML}^\forall$ formula φ_{tree_code} over P such that given a *refinement* $\langle T_r, V_r\rangle$ of a $(k-1)$-initialized full (k,n)-tree code, $\langle T_r, V_r\rangle$ satisfies φ_{tree_code} *iff* $\langle T_r, V_r\rangle$ is a (k,n)-computation tree code. Hence, Property 2 of Lemma 20 follows. We assume that $k > 1$ (the case $k = 1$ being similar). The $\mathsf{RML}^\forall$ formula φ_{tree_code} is defined below.

$$\varphi_{tree\_code} := \Phi_{complete} \wedge \bigwedge_{i=1}^{2n}\big(\Box^{i-1}(\Diamond 0 \wedge \Diamond 1)\big) \wedge \Box^{2n}(\bigwedge_{\ell=1}^{k}\Diamond arg_\ell) \wedge \varphi_{unique} \wedge \varphi_{control} \wedge \varphi_{conf}$$

where the first three conjuncts ensure Property 1 in Definition 8 (in particular, for all $0 \leq i, j \leq 2^n - 1$ and $1 \leq \ell \leq k$, there is some extended block with time position number i, component number ℓ, and tape position number j), while the last three conjuncts given by the formulas φ_{unique}, $\varphi_{control}$, and φ_{conf}, which are defined below, ensure Property 2 in Definition 8. Note that for the latter, we need to ensure that for each $0 \leq i \leq 2^n - 1$, the set of extended blocks having time position number i encodes a (k, n)-configuration. The fulfillment of Property 1 in Definition 8 (enforced by the first three conjuncts in the formula φ_{tree_code}) ensures that Property 2 in Definition 8 is achieved by the following two requirements:

1. *Uniqueness condition:* for all extended TM blocks *ext_bl* and *ext_bl'* having the same component number, the same time position number, and the same tape position number, the following holds: *ext_bl* and *ext_bl'* have the same content. This requirement is enforced by the $\mathsf{RML}^{\forall}$ formula φ_{unique}.

2. *Configuration well-formedness condition:*

   - for each time position number i, there is a unique extended TM block *ext_bl* whose time position number is i and whose content is of the form $(u_-, (q, a), u_+)$. This requirement is enforced by the ML formula $\varphi_{control}$.
   - For all extended TM blocks *ext_bl* and *ext_bl'* having the same component and time position number and such that the associated TM blocks *bl* and *bl'* have position number j and $j + 1$ for some $0 \leq j < 2^n - 1$ (consecutive TM blocks), the following holds: the contents of *ext_bl* and *ext_bl'* are of the form (u_-, u, u_+) and (u, u_+, u_{++}), respectively. Additionally, if $j = 0$ (resp., $j + 1 = 2^n - 1$), then the content of *ext_bl* (resp., *ext_bl'*) is of the form $(\bot, u, u_+)$ (resp., $(u_-, u, \bot)$). This requirement is enforced by the $\mathsf{RML}^{\forall}$ formula φ_{conf}.

$$\varphi_{unique} \;:=\; \forall_r \Big[\Big(\Phi_{complete} \wedge \neg \bigvee_{i=1}^{2n} (\Diamond^i 0 \wedge \Diamond^i 1) \wedge \bigvee_{1 \leq \ell \leq k} \Box^{2n+1} arg_\ell \Big) \longrightarrow \bigvee_{t \in \Lambda} \Box^{2n+2} t \Big]$$

Let $\Lambda(Q)$ be the set of elements in Λ of the form $(u_-, (q, a), u_+)$. Then, the ML formula $\varphi_{control}$ is defined as follows.

$$\begin{aligned} \varphi_{control} \;:=\; \Box^n \Big[& \bigvee_{t \in \Lambda(Q)} \Diamond^{n+2} t \;\wedge\; \neg \bigvee_{t, t' \in \Lambda(Q)} \bigvee_{1 \leq \ell \neq h \leq k} \Big(\Diamond^{n+1} (arg_\ell \wedge \Diamond t) \wedge \Diamond^{n+1} (arg_h \wedge \Diamond t') \Big) \wedge \\ & \neg \bigvee_{t, t' \in \Lambda(Q)} \bigvee_{i=1}^{n} \Big(\Diamond^i (0 \wedge \Diamond^{n-i+2} t) \wedge \Diamond^i (1 \wedge \Diamond^{n-i+2} t') \Big) \Big] \end{aligned}$$

Finally, the $\mathsf{RML}^{\forall}$ formula φ_{conf} is defined as follows.

$$\varphi_{conf} \;:=\; \forall_r\Big(\underbrace{\big(\Phi_{complete} \wedge \neg\big(\bigvee_{i=1}^{n} \Diamond^i 0 \wedge \Diamond^i 1\big) \wedge \bigvee_{1\leq \ell \leq k} \Box^{2n+1} arg_\ell \wedge \varphi_{inc}\big)}_{\text{select two consecutive TM blocks of some component of some } (k,n)\text{-configuration}} \longrightarrow \varphi_{check}\Big)$$

where φ_{inc} and φ_{check} are the ML formulas defined at the end of the proof of Property 1.

Proof of Property 3. We construct in time polynomial in n, k, and the size of the TM $\mathcal{M}$, an $\mathsf{RML}^{\forall}$ formula $\varphi_{faithful}$ over P such that given a (k,n)-computation tree code $\langle T, V\rangle$, $\langle T, V\rangle$ satisfies $\varphi_{faithful}$ *iff* the sequence of (k,n)-configurations encoded by $\langle T, V\rangle$ is faithful to the evolution of $\mathcal{M}$. Hence, Property 3 of Lemma 20 follows. First, we construct two ML formulas φ_{sc} and φ_{sb} that satisfy the following:

- given a refinement $\langle T_r, V_r\rangle$ of a (k,n)-computation tree code $\langle T, V\rangle$, $\langle T_r, V_r\rangle$ satisfies φ_{sc} iff $\langle T_r, V_r\rangle$ "selects" two consecutive (k,n)-configurations encoded by $\langle T, V\rangle$ (i.e., two (k,n) configurations having time position numbers i and $i+1$ for some $0 \leq i < 2^n - 1$);
- given a refinement $\langle T_r, V_r\rangle$ of a (k,n)-computation tree code which "selects" two consecutive (k,n)-configurations $\overrightarrow{C}$ and $\overrightarrow{C}'$, it holds that: for each refinement $\langle T'_r, V'_r\rangle$ of $\langle T_r, V_r\rangle$, $\langle T'_r, V'_r\rangle$ satisfies φ_{sb} iff $\langle T'_r, V'_r\rangle$ "selects" two TM blocks bl and bl' which have the same position and component number, and are associated with $\overrightarrow{C}$ and $\overrightarrow{C}'$, respectively.

$$\varphi_{sc} \;:=\; \Phi_{complete} \wedge \Big(\bigwedge_{i=n+1}^{2n} \Box^{i-1}(\Diamond 0 \wedge \Diamond 1)\Big) \wedge \Big(\Box^{2n} \bigwedge_{1\leq \ell \leq k} \Diamond arg_\ell\Big) \wedge \Diamond 0 \wedge \Diamond 1 \wedge \bigvee_{i=1}^{n}\big(\varphi^i_{cinc} \wedge \neg \bigvee_{j=i+1}^{n} (\Diamond^j 0 \wedge \Diamond^j 1)\big)$$

$$\varphi^i_{cinc} := \begin{cases} \mathtt{true} & \text{if } i = 1 \\ \Box\big(0 \rightarrow (\Box^{i-1} 1 \wedge \bigwedge_{j=1}^{i-2} \Box^j 0)\big) \wedge \Box\big(1 \rightarrow (\Box^{i-1} 0 \wedge \bigwedge_{j=1}^{i-2} \Box^j 1)\big) & \text{otherwise} \end{cases}$$

$$\varphi_{sb} \;:=\; \Phi_{complete} \wedge \bigvee_{j=1}^{n} (\Diamond^j 0 \wedge \Diamond^j 1) \wedge \neg \bigvee_{i=1}^{n} \big(\Diamond^{n+i} 0 \wedge \Diamond^{n+i} 1\big) \wedge \bigvee_{1\leq \ell \leq k} \Box^{2n+1} arg_\ell$$

Now, we describe the construction of the $\mathsf{RML}^{\forall}$ formula $\varphi_{faithful}$:

$$\varphi_{faithful} ::= \forall_r\big(\varphi_{sc} \longrightarrow \bigvee_{(q,a)\in (Q\setminus\{q_{acc},q_{rej}\})\times A} (\psi^{check}_{q,a} \wedge \psi^{faithful}_{q,a})\big)$$

where for a refinement of a (k,n)-computation tree code which selects two (k,n)-configurations $\overrightarrow{C}$ and $\overrightarrow{C}'$ with position numbers i and $i+1$, respectively, we have that: (1) $\psi_{q,a}^{check}$ is a ML formula that checks that $(q,a) \in (Q \setminus \{q_{acc}, q_{rej}\}) \times A$ is the pair (state, scanned cell content) associated with the (k,n)-configuration $\overrightarrow{C}$, and (2) $\psi_{q,a}^{faithful}$ is an $\mathsf{RML}^{\forall}$ formula that uses the ML formula φ_{sb} and checks that $\overrightarrow{C}'$ is the TM successor of $\overrightarrow{C}$.

$$\psi_{q,a}^{check} := \bigvee_{(u_-,(q,a),u_+)\in\Lambda} \bigvee_{i=1}^{n} \Big(\Diamond^i 1 \wedge \Diamond^i (0 \wedge \Diamond^{2n-i+2}(u_-,(q,a),u_+)) \wedge \neg \bigvee_{j=i+1}^{n} (\Diamond^j 0 \wedge \Diamond^j 1) \Big)$$

It remains to construct the $\mathsf{RML}^{\forall}$ formula $\psi_{q,a}^{faithful}$. There are two cases:

- $\delta(q,a) \in Q \times A \times \{\leftarrow, \rightarrow\}$ (ordinary move). Let $\overrightarrow{C} = (C_1(0) \dots C_1(2^{n-1}), \dots, C_k(0) \dots C_k(2^{n-1}))$ be a (k,n)-configuration whose pair (state, scanned cell content) is (q,a). For all $1 \leq \ell \leq k$ and $0 \leq j \leq 2^n - 1$, the 'value' $u_{\ell,j} \in A \cup (Q \times A)$ of the j-th symbol of the ℓth component of the $\overrightarrow{C}$-successor is completely determined by the values $C_\ell(j-1)$, $C_\ell(j)$ and $C_\ell(j+1)$ (taking $C_\ell(j+1)$ for $j = 2^n - 1$ and $C_\ell(j-1)$ for $j = 0$ to be $\perp$). We denote by $next(C_\ell(j-1), C_\ell(j), C_\ell(j+1))$ our expectation for $u_{\ell,j}$ (this function can be trivially obtained from the transition function δ of $\mathcal{M}$). Note that *next* is a function from Λ to $A \cup (Q \times A)$. Thus, we have to check that given a refinement $\langle T_r, V_r \rangle$ of a (k,n)-computation tree code which "selects" two (k,n)-configurations $\overrightarrow{C}$ and $\overrightarrow{C}'$ with time position numbers i and $i+1$ (for some i) respectively (we say that $\overrightarrow{C}$ is the first configuration and $\overrightarrow{C}'$ is the second configuration), and such that (q,a) is the pair (state, scanned cell content) associated with $\overrightarrow{C}$, the following holds: for each refinement of $\langle T_r, V_r \rangle$ which selects two TM blocks bl and bl' such that bl and bl' have the same position and component number, and are associated with $\overrightarrow{C}$ and $\overrightarrow{C}'$, respectively, the content of bl' is of the form $(u'_-, next(u_-, u, u_+), u'_+)$, where (u_-, u, u_+) is the content of bl. Note that this is sufficient and we do not need to impose any constraint between u'_- and u_- and between u'_+ and u_+. Indeed, let j be the position number of bl' and ℓ be the component number of bl'. Then, since $\langle T_r, V_r \rangle$ is a refinement of a (k,n)-computation tree code, our encoding ensures that u'_- (resp., u'_+) represents the value of the $(j-1)$th (resp., $(j+1)$th) symbol – if any – of the ℓth component of $\overrightarrow{C}'$. Moreover, note that for each pair of TM blocks of $\overrightarrow{C}$ and $\overrightarrow{C}'$ having the same position and component number, there is a refinement of $\langle T_r, V_r \rangle$ which selects such a pair. This ensures that the consistency check between u'_- (resp., u'_+) and the associated triple of $\overrightarrow{C}$ is done when the TM block of $\overrightarrow{C}'$ corresponding to u'_- (resp., u'_+) is selected. Thus, the formula $\psi_{q,a}^{faithful}$ is defined as follows:

$$\psi_{q,a}^{faithful} := \forall_r \Big(\varphi_{sb} \longrightarrow \bigvee_{t,(u_-,next(t),u_+)\in\Lambda} \bigvee_{i=1}^{n} \big([\neg \bigvee_{j=i+1}^{n} (\Diamond^j 0 \wedge \Diamond^j 1)] \wedge \underbrace{[\Diamond^i (0 \wedge \Diamond^{2n-i+2} t)]}_{\text{TM block of the first } (k,n)\text{-configuration}} \wedge \underbrace{[\Diamond^i (1 \wedge \Diamond^{2n-i+2}(u_-, next(t), u_+))]}_{\text{TM block of the second } (k,n)\text{-configuration}} \big) \Big)$$

- $\delta(q,a) = h \in \{1, \dots, k\}$ (jump move). For $u \in A \cup (Q \times A)$, the *A-content* $A(u)$ of u is

defined as follows: if u is of the form $(q', a') \in Q \times A$ then $A(u) = a'$; otherwise $A(u) = u$. Let $\overrightarrow{C} = (C_1(0) \dots C_1(2^{n-1}), \dots, C_k(0) \dots C_k(2^{n-1}))$ be a (k,n)-configuration whose pair (state, scanned cell content) is (q, a). Then, the successor $\overrightarrow{C}'$ of $\overrightarrow{C}$ is completed determined by the following constraints:

- For all $1 \leq \ell \leq k$ and $0 \leq j \leq 2^n - 1$, the A-content of the j-th symbol of the ℓ-th component of $\overrightarrow{C}'$ coincides with the A-content of the j-th symbol of the ℓ-th component of $\overrightarrow{C}$ (*A-consistency requirement*).
- the first symbol of the h-th component of $\overrightarrow{C}'$ (recall that $\delta(q, a) = h$) is in $\{q\} \times A$ (*control consistency requirement*).

Thus, we have to check that given a refinement $\langle T_r, V_r \rangle$ of a (k,n)-computation tree code which "selects" two (k,n)-configurations $\overrightarrow{C}$ and $\overrightarrow{C}'$ with time position numbers i and $i+1$ (for some i) respectively, and such that (q, a) is the pair (state, scanned cell content) associated with $\overrightarrow{C}$, the following holds: for each refinement of $\langle T_r, V_r \rangle$ which selects two TM blocks bl and bl' such that bl and bl' have the same position and component number, and are associated with $\overrightarrow{C}$ and $\overrightarrow{C}'$, respectively, the following two conditions are satisfied.

- if (u_-, u, u_+) is the content of bl and (u'_-, u', u'_+) is the content of bl', then $A(u) = A(u')$ (A-consistency requirement).
- If bl and bl' have position number 0 and component number h, then the content of bl' is of the form $(u'_-, (q, b), u'_+)$ (control consistency requirement).

Thus, the formula $\psi_{q,a}^{faithful}$ is defined as follows, where $\Lambda(q)$ denotes the set of elements in Λ of the form $(u_-, (q, b), u_+)$, and for $b \in A$, $\Lambda(b)$ denotes the set of elements $(u_-, u, u_+) \in \Lambda$ such that $A(u) = b$.

$$\psi_{q,a}^{faithful} := \forall_r \Big(\varphi_{sb} \longrightarrow \Big(\underbrace{\Big[\bigvee_{b \in A} \Box^{2n+2} \bigvee_{t \in \Lambda(b)} t \Big]}_{A\text{-}consistency} \wedge$$

$$\underbrace{\Big[\big(\bigwedge_{i=n+1}^{2n} \Box^i 0 \wedge \Diamond^{2n+1} arg_h \big) \longrightarrow \bigvee_{i=1}^{n} \big(\neg \bigvee_{j=i+1}^{n} (\Diamond^j 0 \wedge \Diamond^j 1) \ \wedge \ \Diamond^i 0 \ \wedge \ \Diamond^i (1 \wedge \Diamond^{2n-i+2} \bigvee_{t \in \Lambda(q)} t) \big) \Big]}_{control\ consistency} \Big) \Big)$$

This concludes the proof of Lemma 20. □

Proof of Theorem 17. Theorem 17 directly follows from the following result.

Theorem 21. *One can construct an 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

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

where $\mathsf{Q}_\ell = \exists$ *if* ℓ *is odd, and* $\mathsf{Q}_\ell = \forall$ *otherwise (for all* $1 \leq \ell \leq k$*).*
Additionally, φ *is an* $\mathsf{RML}^\forall$ *formula if* $k = 1$*.*

Proof. First, we consider the case $k = 1$ and then we generalize the proof to the case $k > 1$.

Let $k = 1$, $\varphi_{in\_tree\_code}$ and $\varphi_{faithful}$ be the $\mathsf{RML}^\forall$ formulas satisfying Properties 2–3 of Lemma 20, and φ_{acc} be the ML formula satisfying Property 4 of Lemma 20. Then, the $\mathsf{RML}^\forall$ formula φ (for the case $k = 1$) is given by $\varphi := \varphi_{in\_tree\_code} \wedge \varphi_{faithful} \wedge \varphi_{acc}$. Now, we prove that the construction is correct. We need to show that φ is satisfiable iff $\exists x \in I^{2^n}.\, \mathcal{M}(x)$. Recall that a 0-initialized full (k, n)-tree code is an arbitrary tree structure. Then, the result directly follows from the following chain of equivalences: φ is satisfiable iff $\exists_r \varphi$ is satisfiable iff (by construction and Lemma 20(2–4)) there is a $(1, n)$-computation tree code encoding an accepting $(1, n)$-computation iff $\exists x \in I^{2^n}.\, \mathcal{M}(x)$.

It remains to consider the case $k > 1$. Let $\varphi^1_{init}, \ldots, \varphi^{k-1}_{init}$, $\varphi_{in\_tree\_code}$, and $\varphi_{faithful}$ be the $\mathsf{RML}^\forall$ formulas satisfying Properties 1–3 of Lemma 20, and φ_{acc} be the ML formula satisfying Property 4 of Lemma 20. Define $\varphi_{comp} := \varphi_{in\_tree\_code} \wedge \varphi_{faithful}$. Then, the RML^{k+1} formula φ is defined as follows, where $\widetilde{\mathsf{Q}}_\ell = \exists_r$ and $\mathtt{op}_\ell = \wedge$ if ℓ is odd, and $\widetilde{\mathsf{Q}}_\ell = \forall_r$ and $\mathtt{op}_\ell = \rightarrow$ otherwise (for all $1 \leq \ell \leq k$):

$$\varphi := \widetilde{\mathsf{Q}}_1(\varphi^1_{init}\, \mathtt{op}_1\, \widetilde{\mathsf{Q}}_2(\varphi^2_{init}\ \mathtt{op}_2\ \widetilde{\mathsf{Q}}_3(\varphi^3_{init}\ \mathtt{op}_3\ \ldots\ \mathtt{op}_{k-1}\ \widetilde{\mathsf{Q}}_k(\varphi_{comp}\ \ \mathtt{op}_k\ \ \varphi_{acc})\ldots)))$$

By construction and Lemma 20, it easily follows that φ is an RML^{k+1} formula which can be constructed in time polynomial in n, k, and the size of the TM $\mathcal{M}$. Now, we show that the construction is correct. First, we prove the following claim, where for each $2 \leq \ell \leq k$, we define

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

Claim. Let $1 \leq \ell \leq k-1$, $w_1, \ldots, w_\ell \in I^{2^n}$, and $\langle T, V\rangle$ be an ℓ-initialized full (k, n)-tree code such that $\langle T, V\rangle(w_1, \ldots, w_\ell)$. Then, $\langle T, V\rangle$ satisfies $\varphi_{\ell+1}$ *iff*

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

Proof of the claim. The proof is by induction on $k - \ell$.

Base Step. $\ell = k - 1$. We need to show that $\langle T, V\rangle$ satisfies $\varphi_k = \widetilde{\mathsf{Q}}_k(\varphi_{comp}\ \mathtt{op}_k\ \varphi_{acc})$ iff $\mathsf{Q}_k x_k \in I^{2^n}.\, \mathcal{M}(w_1, \ldots, w_{k-1}, x_k)$. We assume that k is even (the other case being similar). Then, by construction, $\varphi_k = \forall_r(\varphi_{comp} \rightarrow \varphi_{acc})$ and $\mathsf{Q}_k = \forall$.

For the direct implication, assume that $\langle T, V\rangle$ satisfies φ_k. Let $w_k \in I^{2^n}$. We need to prove that $\mathcal{M}(w_1, \ldots, w_{k-1}, w_k)$. Since $\langle T, V\rangle$ is a $k-1$-initialized full (k, n)-tree code such that $\langle T, V\rangle(w_1, \ldots, w_{k-1})$, by Proposition 19(2) there is a refinement $\langle T_r, V_r\rangle$ of $\langle T, V\rangle$ which is a (k, n)-computation tree code encoding the (k, n)-computation from the (k, n)-input $(w_1, \ldots, w_k)$. By Lemma 20(2-3), it follows that $\langle T_r, V_r\rangle$ satisfies $\varphi_{comp} = \varphi_{in\_tree\_code} \wedge \varphi_{faithful}$. Since $\langle T, V\rangle$ satisfies φ_k, we deduce that $\langle T_r, V_r\rangle \models \varphi_{acc}$. Thus, by Lemma 20(4), we obtain that the (k, n)-computation from the (k, n)-input $(w_1, \ldots, w_k)$ is accepting, i.e. $\mathcal{M}(w_1, \ldots, w_k)$.
For the converse implication assume that $\forall x_k \in I^{2^n}.\, \mathcal{M}(w_1, \ldots, w_{k-1}, x_k)$. Let $\langle T_r, V_r\rangle$ be any

refinement of $\langle T, V\rangle$ satisfying $\varphi_{comp} = \varphi_{in\_tree\_code} \wedge \varphi_{faithful}$. We need to show that $\langle T_r, V_r\rangle$ satisfies φ_{acc}. By Lemma 20(2-3), it follows that $\langle T_r, V_r\rangle$ is a (k,n)-computation tree code encoding a (k,n)-computation π. Moreover, since $\langle T_r, V_r\rangle$ is a refinement of $\langle T, V\rangle$ and $\langle T, V\rangle(w_1, \ldots, w_{k-1})$, by Proposition 19(2), there is $w \in I^{2^n}$ such that π represents the (k,n)-computation π from the (k,n)-input $(w_1, \ldots, w_{k-1}, w)$. By hypothesis, $\mathcal{M}(w_1, \ldots, w_{k-1}, w)$, hence π is accepting. Thus, by Lemma 20(4), the result follows.

Induction Step. $1 \leq \ell < k-1$. We assume that $\ell + 1$ is even (the other case being similar). Then, $\varphi_{\ell+1} = \forall_r(\varphi_{init}^{\ell+1} \rightarrow \varphi_{\ell+2})$ and $\mathsf{Q}_{\ell+1} = \forall$. First, assume that $\langle T, V\rangle$ satisfies $\varphi_{\ell+1}$. Let $w_{\ell+1} \in I^{2^n}$. Since $\langle T, V\rangle$ is an ℓ-initialized full (k,n)-tree code such that $\langle T, V\rangle(w_1, \ldots, w_\ell)$, by Proposition 19(1) there must be a refinement $\langle T_r, V_r\rangle$ of $\langle T, V\rangle$ such that $\langle T_r, V_r\rangle$ is an $\ell+1$-initialized full (k,n)-tree code and $\langle T_r, V_r\rangle(w_1, \ldots, w_{\ell+1})$. By Lemma 20(1), $\langle T_r, V_r\rangle \models \varphi_{init}^{\ell+1}$. Since $\langle T, V\rangle$ satisfies $\varphi_{\ell+1}$, we deduce that $\langle T_r, V_r\rangle \models \varphi_{\ell+2}$. Thus, by the induction hypothesis it follows that $\mathsf{Q}_{\ell+2}x_{\ell+2} \in I^{2^n}. \ldots \mathsf{Q}_k x_k \in I^{2^n}. \mathcal{M}(w_1, \ldots, w_{\ell+1}, x_{\ell+2}, \ldots, x_k)$. Since $\mathsf{Q}_{\ell+1} = \forall$ and $w_{\ell+1}$ is arbitrary, we obtain that Condition (1) in the claim holds. For the converse implication, assume that Condition (1) in the claim holds. Let $\langle T_r, V_r\rangle$ be a refinement of $\langle T, V\rangle$ satisfying $\varphi_{init}^{\ell+1}$. We need to show that $\langle T_r, V_r\rangle \models \varphi_{\ell+2}$. By Lemma 20(1), it holds that $\langle T_r, V_r\rangle$ is an $\ell+1$-initialized full (k,n)-tree code. Since $\langle T_r, V_r\rangle$ is a refinement of $\langle T, V\rangle$ and $\langle T, V\rangle(w_1, \ldots, w_\ell)$, by Proposition 19(1) there must be $w_{\ell+1} \in I^{2^n}$ such that $\langle T_r, V_r\rangle(w_1, \ldots, w_{\ell+1})$. Since $\mathsf{Q}_{\ell+1} = \forall$, by hypothesis, it holds that $\mathsf{Q}_{\ell+2}x_{\ell+2} \in I^{2^n}. \ldots \mathsf{Q}_k x_k \in I^{2^n}.\ \mathcal{M}(w_1, \ldots, w_{\ell+1}, x_{\ell+2}, \ldots, x_k)$. Thus, by the induction hypothesis, $\langle T_r, V_r\rangle \models \varphi_{\ell+2}$ holds, and we are done. □

Now, by using the claim above, we show that φ is satisfiable *iff*

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

hence, Theorem 21 follows. Note that $\varphi = \exists_r(\varphi_{init}^1 \wedge \varphi_2)$ and $\mathsf{Q}_1 = \exists$. For the direct implication, let $\langle T, V\rangle$ be a model of φ. Since $\langle T, V\rangle$ is a 0-initialized full (k,n)-tree code, by Lemma 20(1), there is a refinement $\langle T_r, V_r\rangle$ of $\langle T, V\rangle$ which is a 1-initialized full (k,n)-tree code satisfying φ_2. Hence, $\langle T_r, V_r\rangle(w_1)$ for some $w_1 \in I^{2^n}$. Thus, by the claim above for $\ell = 1$ and since $\mathsf{Q}_1 = \exists$, Condition (2) directly follows. For the converse implication, assume that Condition (2) holds. Hence, for some $w_1 \in I^{2^n}$, Condition (1) in the claim with $\ell = 1$ holds. Let $\langle T, V\rangle$ be any 1-initialized full (k,n)-tree code such that $\langle T, V\rangle(w_1)$ (obviously, such a $\langle T, V\rangle$ exists). Then, applying the claim above, we obtain that $\langle T, V\rangle$ satisfies φ_2. Thus, since $\langle T, V\rangle$ is also a refinement of itself, by applying Lemma 20(1), we deduce that $\langle T, V\rangle$ satisfies φ, and we are done. This concludes the proof of Theorem 21. □

6. Concluding remarks

An intriguing question left open is the complexity of satisfiability for *multi-agent* RML [10, 11]. 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 11) 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 $k \geq 1$.[9] Furthermore, since the modal μ-calculus extended with refinement quantifiers (RML^{μ}, for short) is non-elementarily decidable [11], 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.

Acknowledgements

We thank the referees of the journal for their thorough comments. Hans van Ditmarsch is also affiliated to IMSc, Chennai, as an associated researcher. He acknowledges support from ERC starting grant EPS 313360.

References

[1] L. Bozzelli, H. van Ditmarsch, S. Pinchinat, The complexity of one-agent refinement modal logic, in: Proc. of 13th JELIA, LNCS 7519, Springer, 2012, pp. 120–133.

[2] K. Fine, Propositional quantifiers in modal logic, Theoria 36(3) (1970) 336–346.

[3] T. French, Bisimulation quantifiers for modal logic, Ph.D. thesis, University of Western Australia (2006).

[4] A. Visser, Bisimulations, model descriptions and propositional quantifiers, logic Group Preprint Series 161, Department of Philosophy, Utrecht University (1996).

[5] M. Hollenberg, Logic and bisimulation, Ph.D. thesis, University of Utrecht (1998).

[6] S. Pinchinat, A generic constructive solution for concurrent games with expressive constraints on strategies, in: Proc. 5th ATVA, LNCS 4762, Springer, 2007, pp. 253–267.

[7] P. Balbiani, A. Baltag, H. van Ditmarsch, A. Herzig, T. Hoshi, T. D. Lima, 'Knowable' as 'known after an announcement', Review of Symbolic Logic 1(3) (2008) 305–334.

[8] T. French, H. van Ditmarsch, Undecidability for arbitrary public announcement logic, in: C. Areces, R. Goldblatt (Eds.), Advances in Modal Logic 7, College Publications, London, 2008, pp. 23–42, proc. of the seventh conference "Advances in Modal Logic".

[9] H. van Ditmarsch, T. French, Simulation and information, in: Proc. Knowledge Representation for Agents and Multi-Agent Systems, LNAI 5605, Springer, 2009, pp. 51–65.

[10] H. van Ditmarsch, T. French, S. Pinchinat, Future event logic - axioms and complexity, in: Proc. 7th Advances in Modal Logic, Vol. 8, College Publications, 2010, pp. 77–99.

[11] L. Bozzelli, H. van Ditmarsch, T. French, J. Hales, S. Pinchinat, Refinement modal logic, available at http://arxiv.org/abs/1202.3538 (2012).

[9] The complexity of satisfiability for $\mathsf{RML}^{\exists}$ has recently be determined in [23]. The authors also establish results for quantifier alternation, and that the model checking problem for existential RML is PSPACE-complete.

[12] R. Alur, T. Henzinger, O. Kupferman, Alternating-time temporal logic, Journal of the ACM 49(5) (2002) 672–713.

[13] D. Harel, D. Kozen, J. Tiuryn, Dynamic Logic, MIT Press, 2000.

[14] R. Parikh, L. Moss, C. Steinsvold, Topology and epistemic logic, in: Handbook of Spatial Logics, Springer Verlag, 2007, pp. 299–341.

[15] D. Johnson, A catalog of complexity classes, in: Handbook of Theoretical Computer Science, MIT Press, 1990, pp. A:67–161.

[16] C. Papadimitriou, Computational Complexity, Addison Wesley, 1994.

[17] A. Chandra, D. Kozen, L. Stockmeyer, Alternation, Journal of the ACM 28(1) (1981) 114–133.

[18] J. Ferrante, C. Rackoff, A decision procedure for the first order theory of real addition with order, SIAM Journal on Computing 4(1) (1975) 69–76.

[19] T. Rybina, A. Voronkov, Upper bounds for a theory of queues, in: Proc. 30th ICALP, LNCS 2719, Springer, 2003, pp. 714–724.

[20] L. Berman, The complexity of logical theories, Theoretical Computer Science 11 (1) (1980) 71 – 77.

[21] P. Blackburn, M. de Rijke, Y. Venema, Modal Logic, Cambridge University Press, Cambridge, 2001, cambridge Tracts in Theoretical Computer Science 53.

[22] O. Kupferman, M. Vardi, Verification of Fair Transisiton Systems, in: Proc. 8th CAV, LNCS 1102, Springer, 1996, pp. 372–382.

[23] A. Achilleos, M. Lampis, Closing a gap in the complexity of refinement modal logic, `http://arxiv.org/abs/1309.5184` (2013).