Modal Logics with Composition on Finite Forests: Expressivity and Complexity

Bartosz Bednarczyk
TU Dresden & University of Wroclaw

Raul Fervari
FAMAF, Universidad Nacional de Córdoba & CONICET

Stéphane Demri
LSV, CNRS, ENS Paris-Saclay, Université Paris-Saclay

Alessio Mansutti
LSV, CNRS, ENS Paris-Saclay, Université Paris-Saclay

Abstract
We study the expressivity and complexity of two modal logics interpreted on finite forests and equipped with standard modalities to reason on submodels. The logic ML(\()) extends the modal logic K with the composition operator \(\) from ambient logic, whereas ML(*) features the separating conjunction \(\ast\) from separation logic. Both operators are second-order in nature. We show that ML(\()) is as expressive as the graded modal logic GML (on trees) whereas ML(*) is strictly less expressive than GML. Moreover, we establish that the satisfiability problem is Tower-complete for ML(*), whereas it is (only) AExp\_Pol-complete for ML(\()), a result which is surprising given their relative expressivity. As by-products, we solve open problems related to sister logics such as static ambient logic and modal separation logic.

CCS Concepts: • Theory of computation → Modal and temporal logics.

Keywords: modal logic on trees, separation logic, static ambient logic, graded modal logic, expressive power, complexity

ACM Reference Format:

1 Introduction
The ability to quantify over substructures to express properties of a model is often instrumental to perform modular and local reasoning. Two well-known examples are provided by separation logics [28, 35, 42], dedicated to reasoning on pointer programs, and ambient (or more generally, spatial) logics [9, 12, 14, 19], dedicated to reasoning on disjoint data structures. In the realm of modal logics dedicated to knowledge representation, submodel reasoning remains a key ingredient to express the dynamics of knowledge and belief, as done in the logics of public announcement [5, 33, 37], sabotage modal logics [4], refinement modal logics [11] and relation-changing logics [1–3]. Though the models may be of different nature (e.g. memory states for separation logics, epistemic models for logics of public announcement or finite edge-labelled trees for ambient logics), all those logics feature composition operators that enable to compose or decompose substructures in a very natural way.

From a technical point of view, reasoning about submodels requires a global analysis, unlike the local approach for classical modal and temporal logics (typically based on automata techniques [47, 48]). This makes the comparison between those formalisms quite challenging and often limited to a superficial analysis on the different classes of models and composition operators. For instance, the composition operator \(\) in ambient logics decomposes a tree into two disjoint pieces such that once a node has been assigned to one submodel, all its descendants belong to the same submodel. Instead, the separating conjunction \(\ast\) from separation logic decomposes the memory states into two disjoint memory states. Obviously, these and other well-known operators are closely related but no uniform framework investigates exhaustively their relationships in terms of expressive power.

Most of these logics can be easily encoded in monadic second-order logic MSO (or in second-order modal logics [24, 30]). Complexity-wise, if models are tree-like structures, we can then infer decidability thanks to the celebrated Rabin’s theorem [40]. However, most likely, this does not produce the best decision procedures when it comes to solving simple reasoning tasks (e.g. the satisfiability problem of MSO is Tower-complete [43]). Thus, relying on MSO as a common umbrella to capture and understand the differences between those logical formalisms is often not satisfactory.

Our motivations. Our intention in this work is to provide an in-depth comparison between the composition operator \(\) from static ambient logic [12] and the separating conjunction \(\ast\) from separation logics [42] by identifying a common
ground in terms of logical languages and models. As a consequence, we are able to study the effects of having these operators as far as expressivity and complexity are concerned. We aim at defining two logics whose only differences rest on their use of $\mathbf{I}$ and $\ast$ syntactically and semantically (by considering the adequate composition operation). To do so, we pick as our common class of models, the Kripke-style finite trees (actually finite forests, so that the class is closed under taking submodels), which provides an ubiquitous class of structures, extremely well-studied in computer science. For the underlying logical language (i.e. apart from $\mathbf{I}$ or $\ast$), we advocate the use of the standard modal logic $K$ (i.e. to have Boolean connectives and the standard modality $\Diamond$) so that the main operations on the models amount to quantify over submodels or to move along the edges. This framework is sufficiently fundamental to give us the possibility to take advantage of model theoretical tools from modal logics [6, 8, 20]. The benefits of settling a common ground for comparison may lead to further comparisons with other logics and new results.

Our contributions. We introduce $\text{ML}(\mathbf{I})$ and $\text{ML}(\ast)$, two logics interpreted on Kripke-style forest models, equipped with the standard modality $\Diamond$, and respectively with the composition operator [from static ambient logic [12]] and with the separating conjunction $\ast$ from separation logic [42]. Both logical formalisms can state non-trivial properties about submodels, but the binary modalities $\mathbf{I}$ and $\ast$ operate differently: whereas $\ast$ is able to decompose the models at any depth, $\mathbf{I}$ is much less permissive as the decomposition is completely determined by what happens at the level of the children of the current node. We study their expressive power and complexity, obtaining surprising results. We show that $\text{ML}(\mathbf{I})$ is as expressive as the graded modal logic $\text{GML}$ whereas $\text{ML}(\ast)$ is strictly less expressive than $\text{GML}$. Interestingly, this latter development partially reuses the result for $\text{ML}(\mathbf{I})$, hence showing how our framework allows us to transpose results between the two logics. To show that $\text{GML}$ is strictly more expressive than $\text{ML}(\ast)$, we define Ehrenfeucht-Fraïssé games for $\text{ML}(\ast)$. In terms of complexity, the satisfiability problem for $\text{ML}(\mathbf{I})$ is shown $\text{AExp}_{\text{Pol}}$-complete$^1$, interestingly the same complexity as for the refinement modal logic $\text{RML}$ [11] handling a quantifier over refinements (generalising the submodel construction). The $\text{AExp}_{\text{Pol}}$ upper bound follows from an exponential-size model property, whereas the lower bound is by reducing the satisfiability problem for an $\text{AExp}_{\text{Pol}}$-complete team logic [27]. Much more surprisingly, although $\text{ML}(\ast)$ is strictly less expressive than $\text{ML}(\mathbf{I})$, its complexity is much higher (not even elementary). Precisely, we show that the satisfiability problem for $\text{ML}(\ast)$ is Tower-complete. The Tower upper bound is a consequence of [40], whereas hardness is shown by reduction from a Tower-complete tiling problem, adapting substantially the Tower-hardness proof from [7] for second-order modal logic $K$ on finite trees. To conclude, we get the best of our results on $\text{ML}(\mathbf{I})$ and $\text{ML}(\ast)$ to solve several open problems. We relate $\text{ML}(\mathbf{I})$ with an intensional fragment of static ambient logic $\text{SAL}(\mathbf{I})$ from [12] by providing polynomial-time reductions between their satisfiability problems. Consequently, we establish $\text{AExp}_{\text{Pol}}$-completeness of $\text{SAL}(\mathbf{I})$, refute hints from [12, Section 6]. Similarly, we show that the modal separation logic $\text{MSL}(\Diamond^{-1}, \ast)$ from [21] is Tower-complete. Omitted proofs can be found in the technical report on ArXiv.

2 Preliminaries

In this section, we introduce the logics $\text{ML}(\mathbf{I})$ and $\text{ML}(\ast)$ interpreted on tree-like structures equipped with operators to split the structure into disjoint pieces. Due to the presence of such operators, we are required to consider a class of models that is closed under submodels, which we call Kripke-style finite forests (or finite forests for short).

Let $\mathcal{AP}$ be a countably infinite set of atomic propositions. A (Kripke-style) finite forest is a triple $\mathcal{M} = (W, R, V)$ where $W$ is a non-empty finite set of worlds, $V : \mathcal{AP} \to \mathcal{P}(W)$ is a valuation and $R \subseteq W \times W$ is a binary relation whose inverse $R^{-1}$ is functional and acyclic. Then, in particular the graph described by $(W, R)$ is a finite collection of disjoint finite trees (where $R$ encodes the child relation).

We define $R(w) \overset{\mathcal{M}}{=} \{ w' \in W \mid (w, w') \in R \}$. Worlds in $R(w)$ are understood as children of $w$. We inductively define $R^n$: $R^0 \overset{\mathcal{M}}{=} \{ (w, w) \mid w \in W \}; R^{n+1} \overset{\mathcal{M}}{=} \{ (w, w') \mid \exists w'' (w, w'') \in R^n \text{ and (} w', w'' \text{) } \in R \}$. $R^n$ denotes the transitive closure of $R$.

We define operators that chop a finite forest. It should be noted that these operators, as well as the resulting logics, can be cast under the umbrella of the logic of bunched implications $\text{Bi}$ [25, 39], with the exception that we do not explicitly require them to have an identity element (as enforced on the multiplicative operators of $\text{Bi}$, see [25]). Let $\mathcal{M} = (W, R, V)$ and $\mathcal{M}_i = (W_i, R_i, V_i)$ (for $i \in \{1, 2\}$) be three finite forests.

The separation logic composition. We introduce the binary operator $+_\omega$ that performs the disjoint union at the level of parent-child relation. Formally, $\mathcal{M} = \mathcal{M}_1 +_\omega \mathcal{M}_2 \overset{\mathcal{M}}{=} R_1 \cup R_2 = R, W_1 = W_2 = W, V_1 = V_2 = V$.

This is the composition used in separation logic [21, 42]. The figure below depicts possible instances for $\mathcal{M, M}_1$ and $\mathcal{M}_2$.

The ambient logic composition. We introduce the operator $+_w$, where $w \in W$, that constrains further $+$: $\mathcal{M} = \mathcal{M}_1 +_w \mathcal{M}_2 \overset{\mathcal{M}}{=} \mathcal{M} = \mathcal{M}_1 + \mathcal{M}_2$ and $R^+_w (w') = R^+_w (w')$ holds for all $i \in \{1, 2\}$ and $w' \in R_i (w)$. $\mathcal{M}$ is a disjoint union between $\mathcal{M}_1$ and $\mathcal{M}_2$ except that, as soon as $w' \in R_i (w)$, the whole subtree of $w'$ in $R$ belongs

---

$^1$ Problems in $\text{AExp}_{\text{Pol}}$ are decidable by an alternating Turing machine working in exponential-time and using polynomially many alternations [10].
to $\mathcal{M}_1$, like the composition in ambient logic [12]. Below, we illustrate a model decomposed with $+_w$.

We say that $\mathcal{M}_1$ is a submodel of $\mathcal{M}_2$, written $\mathcal{M}_1 \subseteq \mathcal{M}_2$ if there is $\mathcal{M}_2$ such that $\mathcal{M} = \mathcal{M}_1 + \mathcal{M}_2$.

**Modal logics on trees.** The logic ML( ) enriches the modal logic K (a.k.a. ML) with a binary connective $\vdash$, called composition operator, that admits submodel reasoning via the operator $+_w$. Similarly, ML( ) enriches ML with the connective $\ast$, called separating conjunction (or star) that admits submodel reasoning via the operator $\vdash$. Both connectives $\vdash$ and $\ast$ are understood as binary modalities. As we show throughout the paper, ML( ) and ML( ) are strongly related to the graded modal logic GML [20]. For conciseness, let us define all these logics by considering formulae that contain all of their ingredients. These formulae are built from

$$\varphi := \top | p | \varphi \land \varphi | \neg \varphi | \Diamond \varphi | \Diamond_{\geq k} \varphi | \varphi \ast \varphi | \varphi \varphi,$$

where $p \in \text{AP}$ and $k \in \mathbb{N}$ (encoded in binary). A pointed forest $(\mathcal{M}, w)$ is a finite forest $\mathcal{M} = (W, R, V)$ together with a world $w \in W$. The satisfaction relation $\models$ is defined as follows (standard clauses for $\land$, $\neg$, and $\top$ are omitted):

- $\mathcal{M}, w \models \varphi \iff v \in V(\varphi)$;
- $\mathcal{M}, w \models \Diamond \varphi \iff \exists w' \in R(w) \text{ s.t. } \mathcal{M}, w' \models \varphi$;
- $\mathcal{M}, w \models \Diamond_{\geq k} \varphi \iff \{w' \in R(w) | \mathcal{M}, w' \models \varphi\} \geq k$;
- $\mathcal{M}, w \models \varphi_1 \ast \varphi_2 \iff \exists \mathcal{M}_1, \mathcal{M}_2 \text{ s.t. } \mathcal{M} = \mathcal{M}_1 + \mathcal{M}_2$;
- $\mathcal{M}, w \models \varphi_1$, $\mathcal{M}_1, w \models \varphi_2$, $\mathcal{M}_2, w \models \varphi_2$.

The formulae $\varphi \vdash \psi$, $\varphi \land \psi$ and $\bot$ are defined as usual. We use the following standard abbreviations: $\Box \varphi \iff \Diamond \neg \varphi$, $\Diamond_{\leq k} \varphi \iff \Diamond_{\leq k+1} \varphi$ and $\Diamond_{\geq k} \varphi \iff \Diamond_{\geq k} \varphi \land \Diamond_{\leq k} \varphi$. We write $\text{size}(\varphi)$ to denote the size of $\varphi$ with a tree representation of formulae and with a reasonably succinct encoding of atomic formulae. Besides, we write $\text{md}(\varphi)$ to denote the modal degree of $\varphi$ understood as the maximal number of nested unary modalities (i.e. $\Diamond$ or $\Diamond_{\geq k}$) in $\varphi$. Similarly, the graded rank $gr(\varphi)$ of $\varphi$ is defined as $\max\{|k| \Diamond_{\geq k} \psi \in \text{sub}(\varphi)\} \cup \{0\}$, where sub( ) is the set of all subformulae of $\varphi$.

Given the formulae $\varphi$ and $\psi$, $\varphi \equiv \psi$ denotes that $\varphi$ and $\psi$ are logically equivalent; i.e., for every pointed forest $(\mathcal{M}, w)$, $\mathcal{M}, w \models \varphi$ iff $\mathcal{M}, w \models \psi$. For instance (k $\geq 1$ and p $\in$ AP):

1. $\Diamond \varphi \equiv \Diamond_{\geq 1} \varphi$;
2. $(\Box \bot \Box \bot) \equiv (\Box \bot \ast \Box \bot)$;
3. $\Diamond_{\geq k} p \equiv \Diamond_{\geq k} \Diamond_{\geq k} \cdot \ast \Diamond_{\geq k} p$, $k$ times;
4. $\Diamond_{\geq k} \varphi \equiv \Diamond_{\geq k} \varphi \ast \Diamond_{\geq k} \varphi$, $k$ times.

The modal logic ML is the logic restricted to formulae with the unique modality $\Diamond$ [8]. Similarly, the graded modal logic GML is restricted to the graded modalities $\Diamond_{\geq k}$ [20]. We introduce the modal logics ML( ) and ML( ), which are restricted to the suites of modalities $\{\Diamond, \ast\}$ and $\{\Diamond, \vdash\}$, respectively. The two equivalences (3) and (4) already shed some light on ML( ) and ML( ); the two logics are similar when it comes to their formulae of modal degree one.

**Lemma 2.1.** Let $\varphi$ be a formula in ML( ) with $\text{md}(\varphi) \leq 1$. Then, $\varphi \equiv \varphi \ast \varphi$ is the formula in ML( ) obtained from $\varphi$ by replacing every occurrence of $\vdash$ by $\ast$.

However, as shown by the non-equivalence (2), it is unclear how the two logics compare when it comes to formulae of modal degree greater than one. Indeed, since $\mathcal{M} = \mathcal{M}_1 + \mathcal{M}_2$ implies $\mathcal{M} = \mathcal{M}_1 + \mathcal{M}_2$, but not vice-versa, the separating conjunction $\ast$ is more permissive than the operator $\vdash$. However, further connections between the two operators can be easily established. We let introduce the auxiliary operator $\Phi$ defined as $\Box \varphi \equiv \varphi \ast \Diamond \bot$. Formally,

$$\mathcal{M}, w \models \varphi \iff \text{there is } R' \subseteq R \text{ s.t. } R'(w) = R(w) \text{ and } R'(w) \backslash R(w).$$

**Lemma 2.2.** Let $\varphi, \psi \in \text{GML}$. We have $\varphi \ast \psi \equiv \varphi \Phi \psi$.

Unlike $\vdash$, when $\ast$ splits a finite forest $\mathcal{M}$ into $\mathcal{M}_1$ and $\mathcal{M}_2$, it may disconnect in both submodels worlds that are otherwise reachable, from the current world, in $\mathcal{M}$. Applying $\Phi$ before $\ast$ allows us to imitate this behaviour. Indeed, even though $\vdash$ preserves reachability in either $\mathcal{M}_1$ or $\mathcal{M}_2$, $\Phi$ deletes part of $\mathcal{M}$, making some world inaccessible. This way of expressing the separating conjunction allows us to reuse some methods developed for ML( ) in order to study ML( )

**The logic QK'.** Both ML( ) and ML( ) can be seen as fragments of the logic QK', which in turn is known to be a fragment of monadic second-order logic on trees [7]. The logic QK' extends ML with second-order quantification and is interpreted on finite trees. Its formulae are defined according to the following grammar:

$$\varphi := p \mid \Diamond \varphi \mid \varphi \land \varphi \mid \neg \varphi \mid \exists p \varphi.$$

Given $\mathcal{M} = (W, R, V)$ and $w \in W$, the satisfaction relation $\models$ of ML is extended as follows:

$$\mathcal{M}, w \models \exists p \varphi \text{ iff } \exists W' \subseteq W \text{ s.t. } (W, R, V[p \leftarrow W'], w) \models \varphi.$$

One can show logspace reductions from ML( ) and ML( ) to QK', by simply reinterpreting the operators $\ast$ and $\vdash$ as restrictive forms of second-order quantification, and by relativising $\Diamond$ to appropriate propositional symbols in order to capture the notion of submodel (details are omitted).

**Satisfiability problem.** The satisfiability problem for a logic $\mathcal{L}$, written Sat($\mathcal{L}$), takes as input a formula $\varphi$ in $\mathcal{L}$ and checks whether there is a pointed forest $(\mathcal{M}, w)$ such that $\mathcal{M}, w \models \varphi$.

Note that any $\mathcal{L}$ among ML, GML, ML( ) or ML( ) has the tree model property, i.e. any satisfiable formula is also satisfied in some tree structure. The problems Sat(ML) and
Sat(GML) are known to be PSPACE-complete, see e.g. [8, 29, 44, 45], and therefore Sat(ML(1)) and Sat(ML(*)) are PSPACE-hard. As an upper bound, by Rabin’s theorem [40], the satisfiability problem for QK* is decidable in Tower, which transfers directly to Sat(ML(1)) and Sat(ML(*)).

Expressive power. Given two logics $\mathcal{L}_1$ and $\mathcal{L}_2$, we say that $\mathcal{L}_2$ is at least as expressive as $\mathcal{L}_1$ (written $\mathcal{L}_1 \leq \mathcal{L}_2$) whenever for every formula $\varphi$ of $\mathcal{L}_1$, there is a formula $\psi$ of $\mathcal{L}_2$ such that $\varphi \equiv \psi$. $\mathcal{L}_1 \approx \mathcal{L}_2$ denotes that $\mathcal{L}_1$ and $\mathcal{L}_2$ are equally expressive, i.e. $\mathcal{L}_1 \leq \mathcal{L}_2$ and $\mathcal{L}_2 \leq \mathcal{L}_1$. Lastly, $\mathcal{L}_1 \prec \mathcal{L}_2$ denotes that $\mathcal{L}_2$ is strictly more expressive than $\mathcal{L}_1$, i.e. $\mathcal{L}_1 \leq \mathcal{L}_2$ and $\mathcal{L}_1 \not\approx \mathcal{L}_2$.

The equivalence (1) recalls us that $\mathcal{L} \times \mathcal{GML}$ [20]. From the equivalence (4), we get $\mathcal{GML} \leq \mathcal{ML}(1)$.

3 ML(1): Expressiveness and Complexity

In this section, we study the expressive power of ML(1) and the complexity of Sat(ML(1)). We show constructively that $\mathcal{ML}(1) \leq \mathcal{GML}$, hence proving $\mathcal{ML}(1) \equiv \mathcal{GML}$. Next, we show that Sat(ML(1)) is AEExpPol-complete. The upper bound is achieved by proving an exponential-size model property. The lower bound is by reduction from the satisfiability problem for propositional team logic [27, Thm. 4.9].

3.1 ML(1) is not more expressive than GML

Establishing $\mathcal{ML}(1) \leq \mathcal{GML}$ amounts to show that given $\varphi_1$, $\varphi_2$ in GML, one can construct $\psi$ in GML such that $\varphi_1 \models \varphi_2 \equiv \psi$. For instance, a simple case analysis yields the equivalence $(p \lor \diamond_{\geq 3} r) \models (q \lor \diamond_{\leq 5} q) \equiv (p \lor \diamond_{\geq 3} r)$. With this property, the general algorithm consists in iteratively replacing innermost subformulae of the form $\varphi_1 | \varphi_2$ by a counterpart in GML, allowing us to eliminate all the occurrences of $\mathcal{L}$ and obtain an equivalent formula in GML. The base case involves subformulae $\varphi_1$ and $\varphi_2$ in ML (a fragment of GML).

Let us provide a few definitions. Let $\varphi$ be a formula in GML. We write $\maxpol(\varphi)$ to denote the set of subformuale $\psi$ of $\varphi$ that are maximal and modality-free, i.e.

1. $\psi$ is modality-free: it does not contain modalities $\diamond_{\geq k}$ and one of its occurrences is not in the scope of $\diamond_{\geq k}$;
2. $\psi$ is maximal: one of its occurrences does not belong to a larger modality-free subformula of $\varphi$.

For instance, $\maxpol((p \lor \diamond_{\geq 3} r) \land (q \lor p)) = \{p, q \lor p\}$. Similarly, $\maxpol(\varphi)$ denotes the set of subformulae $\psi$ of $\varphi$ such that $\psi$ is of the form $\diamond_{\geq k} \psi'$ and one of its occurrences in $\varphi$ is not in the scope of graded modalities $\diamond_{\geq k}$. For instance, $\maxpol((p \lor \diamond_{\geq 3} r) \land (q \lor \diamond_{\leq 5} \diamond_{\geq 2} q)) = \{\diamond_{\geq 3} r, \diamond_{\geq 5} \diamond_{\geq 2} q\}$. Every formula $\varphi$ in GML is a Boolean combination of formulae from $\maxpol(\varphi) \cup \maxpol(\varphi)$. Lastly, $\varphi$ is in good shape if the properties (1) and (2) hold below:

1. $\maxpol(\varphi) \subseteq \{\bot, \top\}$. Consequently, every propositional variable in $\varphi$ occurs in the scope of a graded modality;
2. For all $\diamond_{\geq k} \psi, \diamond_{\geq k} \psi'$ in $\maxpol(\varphi)$ with $\psi \neq \psi'$, the conjunction $\psi \land \psi'$ is unsatisfiable.

Let $\varphi_1$ and $\varphi_2$ be GML formulae. First, we show that when $\varphi_1 \land \varphi_2$ is in good shape, there is a GML formula $\psi$ such that $\varphi_1 \models \varphi_2 \equiv \psi$. To do so, we take a slight detour through Presburger arithmetic (PA), see e.g. [26, 38]. Given two formulae $\varphi_1, \varphi_2$ in GML, we will characterise the formula $\varphi_1 | \varphi_2$ by using arithmetical constraints for the number of successors. Then, we will take advantage of basic properties of PA in order to eliminate quantifiers, and obtain a GML formula. Below, the variables $x, y, z, \ldots$ are decorated and occurring in formulae, are from PA and therefore they are interpreted by natural numbers.

Let $\varphi$ be in GML s.t. $\maxpol(\varphi) \subseteq \{\bot, \top\}$ and $\{\psi_1, \ldots, \psi_n\}$ contains the set $\{\psi_1 | \diamond_{\geq k} \psi \in \maxpol(\varphi)\}$. We define formulae in PA that state constraints about the number of children satisfying a formula $\psi_j$. The variable $x_j$ is intended to be interpreted as the number of children satisfying $\psi_j$. We write $\varphi^{PA}(x_1, \ldots, x_n)$ to denote the arithmetical formula obtained from $\varphi$ by replacing with $x_j \geq k$ every occurrence of $\diamond_{\geq k} \psi$ that is in the scope of a graded modality. For instance, assuming that $\varphi = \diamond_{\geq 5} (p \land q) \lor \diamond_{\geq 4} \neg \psi$, the expression $\varphi^{PA}(x_1, x_2)$ denotes the formula $x_1 \geq 5 \lor \neg (x_2 \geq 4)$.

Let $\varphi_1, \varphi_2$ be GML formulae such that $\varphi_1 \land \varphi_2$ is in good shape and $\{\psi_1, \ldots, \psi_n\} = \{\psi_1 | \diamond_{\geq k} \psi \in \maxpol(\varphi_1 \land \varphi_2)\}$. We consider the formula $[\varphi_1, \varphi_2]^{PA}$ in PA defined below:

$$[\varphi_1, \varphi_2]^{PA} \equiv \exists y_1, y_1', \ldots, y_n, y_n' (\land_{j=1}^n x_j = y_j') \land
\varphi_1^{PA}(y_1', \ldots, y_n') \land \varphi_2^{PA}(y_1', \ldots, y_n').$$

The formula $[\varphi_1, \varphi_2]^{PA}$ states that there is a way to divide the children in two distinct sets and each set allows to satisfy $\varphi_1^{PA}$ or $\varphi_2^{PA}$, respectively. As PA admits quantifier elimination [16, 38, 41], there is a quantifier-free formula $\chi$ equivalent to $[\varphi_1, \varphi_2]^{PA}$ and its free variables are among $x_1, \ldots, x_n$. A priori, the atomic formulae of $\chi$ may be not of the simple form $x_j \geq k$ (e.g. ‘modulo constraints’ or constraints of the form $\sum a_i x_j \geq k$ may be involved). However, if the atomic formulae of $\chi$ are restricted to expressions of the form $x_j \geq k$, then we write $\chi^{GML}$ to denote the GML formula obtained from $\chi$ by replacing every occurrence of $x_j \geq k$ by $\diamond_{\geq k} \psi_j$.

**Lemma 3.1.** Let $\varphi_1, \varphi_2$ be in GML such that $\varphi_1 \land \varphi_2$ is in good shape. $[\varphi_1, \varphi_2]^{PA}$ is equivalent to a quantifier-free PA formula $\chi$ whose atomic formulae are only of the form $x_j \geq k$. Moreover, $\varphi_1 | \varphi_2 \equiv \chi^{GML}$ and $\text{gr}(\chi^{GML}) \leq \text{gr}(\varphi_1) + \text{gr}(\varphi_2)$.

The bound on $\text{gr}(\chi^{GML})$ stated in this key lemma is essential to obtain an exponential bound on the smallest model satisfying a formula in ML(1) (see Section 3.2). Thanks to Lemma 3.1, we can show that GML is closed under the operator $|$ by reducing the occurrences of this operator to formulae in good shape. In particular, we show that given two arbitrary formulae $\varphi_1$ and $\varphi_2$ in GML, $\varphi_1 | \varphi_2$ is equivalent to a disjunction of formulae of the form $[\psi_1 | \psi_2] \land \chi$, where $\chi$ is a Boolean combination of atomic propositions and $\psi_1 \land \psi_2$ is in good shape (hence $[\psi_1 | \psi_2]$ is equivalent to a formula in GML...
by Lemma 3.1). This is shown syntactically: atomic propositions are dealt with by propositional reasoning, whereas to produce $\psi_1$ and $\psi_2$ we use axioms from GML [6] and rely on the following equivalences:

(guess) $\vdash_{\geq k} \phi \equiv \vdash_{\geq k} ((\phi \land \psi) \lor (\phi \land \neg \psi))$;

(good dist) if $\phi \land \psi$ unsat., $\vdash_{\geq k} (\phi \lor \psi \equiv \vdash_{\geq k} (\phi \land \neg \psi))$;

(disc) $\vdash (\phi \lor \psi) \rightarrow (\phi \lor (\psi \lor \chi))$.

Notice that the conjunction of $\phi \land \psi$ and $\phi \land \neg \psi$ from (guess) is trivially unsatisfiable, allowing us to use (good dist). As GML is shown to be closed under the operator $\vdash$, we conclude.

**Theorem 3.2.** $ML(\{\}) \leq$ GML. Therefore, $ML(\{\}) \cong$ GML.

To prove $ML(\{\}) \leq$ GML, we iteratively put subformulae in good shape and apply Lemma 3.1. This is done several times, potentially causing an exponential blow-up each time a formula is transformed. To provide an optimal complexity upper bound, we need to tame this combinatorial explosion.

### 3.2 AExpPol-completeness

In order to show that $Sat(ML(\{\}))$ is in $AExpPol$, the main ingredient is to show that given $\varphi$ in $ML(\{\})$, we build $\varphi'$ in GML such that $\varphi' \equiv \varphi$ and the models for $\varphi'$ (if any) do not require a number of children per node more than exponential in $size(\varphi)$. The proof of Theorem 3.2 needs to be refined to improve the way $\varphi'$ is computed. In particular, this requires a strategy for the application of the equivalences used to put a formula in good shape.

We need to introduce a few more simple notions. Let $\varphi$ be a GML formula with $maxGML(\varphi) = \{\vdash_{\geq k_1} \psi_1, \ldots, \vdash_{\geq k_n} \psi_n\}$. We define $bd(0, \varphi) \triangleq k_1 + \ldots + k_n$. For all $m \geq 0$, we define $bd(m + 1, \varphi) \triangleq \max\{bd(m, \psi) \mid \vdash_{\geq k} \psi \in maxGML(\varphi)\}$. Hence, $bd(m, \varphi)$ can be understood as the maximal $bd(0, \psi)$ for some subformula $\psi$ occurring at the modal depth $m$ within $\varphi$. We write $maxbd(\varphi)$ for the value $\max\{bd(m, \varphi) \mid m \in [0, \text{md}(\varphi)]\}$. If $\varphi$ is satisfiable, we can use $maxbd(\varphi)$ to obtain a bound on the smallest model satisfying it, as stated in Lemma 3.3 below.

**Lemma 3.3.** Every satisfiable $\varphi$ in GML is satisfiable by a pointed forest with at most $maxbd(\varphi)^{maxbd(\varphi)+1}$ worlds.

To show that $ML(\{\})$ has the exponential-size model property, we establish that given $\varphi$ in $ML(\{\})$, there is $\varphi'$ in GML such that $\varphi' \equiv \varphi$, $md(\varphi') \leq md(\varphi)$ and $maxbd(\varphi')$ is exponential in $size(\varphi)$. First, we consider the fragment $F$ of $ML(\{\})$:

$\varphi := \vdash_{\geq k} \psi \lor \neg \psi \land \neg \psi$.

When $p \equiv AP$ and $\vdash_{\geq k} \psi$ is a formula in GML (abusingly assumed in $ML(\{\})$ but we know $GML \leq ML(\{\})$), given $\varphi$ in $ML(\{\})$ or in $F$, we write $cd(\varphi)$ to denote its composition degree, i.e. the maximal number of imbrications of $[\varphi]$ in $\varphi$. We extend the notion of $bd$ to formulae in $F$, so that $bd(m, \varphi) = bd(m, \varphi)[\leftarrow \land]$.

Let $\varphi$ be in $F$ such that $maxGML(\varphi) = \{\vdash_{\geq k_1} \chi_1, \ldots, \vdash_{\geq k_n} \chi_n\}$. The key step to show the exponential-size model property essentially manipulates the formulae in $maxGML(\varphi)$ in order to produce equivalent formulae $\psi_1, \ldots, \psi_n$, so that for all distinct $i$ and $j$, $\psi_i \land \psi_j$ is in good shape. Moreover, by replacing in $\varphi$ every $\vdash_{\geq k} \chi_i$ with the equivalent formula $\psi_i$, we only witness an exponential blow-up on $bd(0, \varphi)$, whereas for every $m > 1$, $bd(m, \varphi)$ remains polynomially bounded by the $bd$ of the original formula. With the bound on the graded rank found in Lemma 3.1, we derive Lemma 3.4.

**Lemma 3.4.** Let $\varphi$ be a formula of the fragment $F$ such that $maxGML(\varphi) = \{\vdash_{\geq k_1} \chi_1, \ldots, \vdash_{\geq k_n} \chi_n\}$ and $k = \max\{k_1, \ldots, k_n\}$. There is a GML formula $\psi$ such that $\varphi \equiv \psi$ and

1. $md(\psi) \leq md(\varphi)$
2. $bd(0, \psi) \leq \bar{k} \times 2^{n+cd(\varphi)}$
3. $bd(1, \psi) \leq n \times bd(1, \varphi)$
4. $\forall m \geq 2, \text{bd}(m, \psi) = \text{bd}(m, \varphi)$.

In the proof of Lemma 3.4, a first step essentially consists in applying multiple times (guess) in order to derive, for every $i \in [1, n]$, an equivalence $\vdash_{\geq k_i} \chi_i \equiv \psi_i'$ where $\psi_i' \equiv \vdash_{\geq k_i} \psi_i \lor \psi_i \land \neg \psi_i$. Notice that the conjunction of $\psi_i'$ and $\psi_i \land \neg \psi_i$ from (guess) is trivially unsatisfiable, allowing us to use (good dist). As GML is shown to be closed under the operator $\vdash$, we conclude.

**Theorem 3.6.** $Sat(ML(\{\}))$ is in $AExpPol$.

The (standard) proof consists in observing that to check the satisfiability status of $\varphi$ in $ML(\{\})$, first guess a pointed forest of exponential-size (thanks to Lemma 3.5) and check that it satisfies $\varphi$. This can be done in exponential-time using an alternating Turing machine with a linear amount of alternations (between universal states and existential states) by viewing $ML(\{\})$ as a fragment of MSO.

It remains to establish $AExpPol$-hardness. We provide a logspace reduction from the satisfiability problem for the team logic PL[-] shown $AExpPol$-complete in [27, Thm. 4.9].
PL[-] formulae are defined by the following grammar:
\[ \varphi := p \mid \neg p \mid \varphi \land \varphi \mid \neg \varphi \mid \varphi \lor \varphi, \]
where \( p \in AP \) and the connectives \( \neg \) and \( \lor \) are dotted to avoid confusion with those of ML(1). PL[-] is interpreted on sets of (Boolean) propositional valuations over a finite subset of AP. They are called teams and are denoted by \( \mathcal{I}, \mathcal{I}_1, \ldots \). A model for \( \varphi \) is a team \( \mathcal{I} \) over a set of propositional variables including those occurring in \( \varphi \) and such that \( \mathcal{I} \models \varphi \) with:
\[ \mathcal{I} \models p \iff \text{for all } v \in \mathcal{I}, \text{we have } o(p) = 1; \]
\[ \mathcal{I} \models \varphi_1 \land \varphi_2 \iff \mathcal{I} \models \varphi_1 \land \mathcal{I} \models \varphi_2. \]
The connectives \( - \) and \( \land \) are interpreted as the classical negation and conjunction, respectively. Notice that, in the clause for \( \lor \), the teams \( \mathcal{I}_1 \) and \( \mathcal{I}_2 \) are not necessarily disjoint.

Let us discuss the reduction from Sat(PL[-]) to Sat(ML(1)). A direct encoding of a team \( \mathcal{I} \) into a pointed forest \( (\mathcal{W}, w) \) consists in having a correspondence between the propositional valuations in \( \mathcal{I} \) and the propositional valuations of the children of \( w \). This would work fine if there were no mismatch between the semantics for \( 1 \) (disjointness of the children) and the one for \( v \) (disjointness not required). To handle this, when checking the satisfaction of \( \varphi \) in PL[-] with \( n \) occurrences of \( \lor \), we impose that if a propositional valuation occurs among the children of \( w \), then it occurs in at least \( n + 1 \) children. This property must be maintained after applying \( \lor \) several times, always with respect to the number of occurrences of \( \lor \) in the subformula of \( \varphi \) that is evaluated.

Non-disjointness of the teams is encoded by carefully separating the children of \( w \) having identical valuations.

We now formalise the reduction. Assume that we wish to translate \( \varphi \) from PL[-], written with atomic propositions in \( P = \{p_1, \ldots, p_m\} \) and containing at most \( n \) occurrences of the operator \( \lor \). We introduce a set \( Q = \{q_1, \ldots, q_{n+1}\} \) of auxiliary propositions disjoint from \( P \). The elements of \( Q \) are used to distinguish different copies of the same propositional valuation of a team. Thus, with respect to a pointed forest \( (\mathcal{W}, w) \), we require each child of \( w \) to satisfy exactly one element of \( Q \). This can be done with the formula
\[ \text{uni}(Q) \triangleq \Box \bigwedge_{i \neq j \in [1, n+1]} \neg(q_i \land q_j) \land \bigvee_{i \in [1, n+1]} q_i. \]

We require that if a child of \( w \) satisfies a propositional valuation over (elements in) \( P \), then there are \( n + 1 \) children satisfying that valuation over \( P \), each of them satisfying a distinct symbol in \( Q \). So, every valuation over \( P \) occurring in some child of \( w \), occurs at least in \( n + 1 \) children of \( w \). However, as the translation of the operator \( \lor \) modifies the set of copies of a propositional valuation, this property must be extended to arbitrary subsets of \( Q \). Given \( \emptyset \neq X \subseteq [1, n+1] \), we require that for all \( k \neq k' \in X \), if a children of \( w \) satisfies \( q_k \), then there is a child satisfying \( q_{k'} \) with the same valuation on \( P \). The formula \( \text{cp}(X) \) below does the job:
\[ \bigwedge_{k \neq k' \in X} \neg(q_k \land q_{k'} \land \neg(T \mid \bigwedge_{i \in [1, n]} \neg(p_j \Rightarrow \neg p_j))). \]

Lastly, before defining the translation map \( \tau \), we describe how different copies of the same propositional valuation are split. We introduce two auxiliary choice functions \( c_1 \) and \( c_2 \) that take as arguments \( X \subseteq [1, n+1] \), and \( n_1, n_2 \in \mathbb{N} \) with \( |X| \geq n_1 + n_2 + 1 \) such that for each \( i \in \{1, 2\} \), we have \( c_i(X, n_1, n_2) \subseteq X, |c_i(X, n_1, n_2)| \geq n_i \). Moreover, \( c_1(X, n_1, n_2) \subseteq c_2(X, n_1, n_2) \subseteq X \). The maps \( c_1 \) and \( c_2 \) are instrumental to decide how to split \( X \) into two disjoint subsets respecting basic cardinality constraints. The translation map \( \tau \) is designed as follows (\( 0 \neq X \subseteq [1, n+1] \)):
\[ \tau(p, X) \triangleq \Box((\bigvee_{j \in X} q_j) \Rightarrow p); \]
\[ \tau(\neg p, X) \triangleq \Box((\bigvee_{j \in X} q_j) \Rightarrow \neg p); \]
\[ \tau(q_1 \land q_2, X) \triangleq \tau(q_1, X) \land \tau(q_2, X); \]
\[ \tau(\neg \varphi, X) \triangleq \neg \tau(\varphi, X); \]
\[ \tau(\varphi_1 \lor q_2, X) \triangleq \tau(\varphi_1, X) \land \text{cp}(X) \land \tau(q_2, X) \land \text{cp}(X_2), \]
where \( (\{j\}|X) \) is greater or equal to the number of occurrences of \( \lor \) in \( q_2 \) plus one; \( (ii) \) given \( n_1, n_2 \) such that \( n_1 \) (resp. \( n_2 \)) is the number of occurrences of \( \lor \) in \( q_1 \) (resp. \( q_2 \)) plus one, for each \( i \in \{1, 2\} \) we have \( c_i(X, n_1, n_2) = X_i \).

Lemma 3.7 below guarantees that starting with a linear number of children with the same propositional valuation is sufficient to encode \( \lor \) within ML(1).

**Lemma 3.7.** Let \( \varphi \in \text{PL[-]} \) with \( n \) occurrences of \( \lor \) and built upon \( p_1, \ldots, p_m \). Then, \( \varphi \) is satisfiable iff so is
\[ \text{uni}(q_1, \ldots, q_{n+1}) \land \text{cp}(\{1, n+1\}) \land \tau(\varphi, \{1, n+1\}). \]

The ML(1) formula involved in Lemma 3.7 has modal depth one. By Theorem 3.6, Sat(ML(1)) is AEExpPol-complete even restricted to formulae of modal depth at most one.

**Corollary 3.8.** Sat(ML(1)) is AEExpPol-complete.

As we show in the next section, the complexity of ML(*) does not collapse to modal depth one: Sat(ML(*)) restricted to formulae of modal depth \( k \) is exponentially easier than Sat(ML(*)) restricted to formulae of modal depth \( k + 1 \).

### 4 ML(*) is Tower-complete

We show that Sat(ML(*)) is Tower-complete, i.e. complete for the class of all problems of time complexity bounded by a tower of exponentials whose height is an elementary function \([43]\). Given \( k, n \geq 0 \), we inductively define the tetrational function \( t \) as \( t(0, 0) \triangleq n \) and \( t(k, n) = 2^{t(k,n)} \). Intuitively, \( t(k, n) \) defines a tower of exponentials of height \( k \). By \( k\text{-NExpTime} \), we denote the class of all problems decidable with a nondeterministic Turing machine (NTM) of working time \( O(t(k, p(n))) \) for some polynomial \( p(n) \), on each input of length \( n \). To show Tower-hardness, we design a uniform elementary reduction allowing us to get \( k\text{-NExpTime} \) hardness for all \( k \) greater than a certain (fixed) integer. In our case, we achieve an exponential-space reduction from the \( k\text{-NExpTime} \) variant of the tiling problem, for all \( k \geq 2 \).

The tiling problem \( 112\text{e}_k \) takes as input a triple \( \overline{T} = (\mathcal{T}, \mathcal{H}, \mathcal{V}) \) where \( \mathcal{T} \) is a finite set of tile types, \( \mathcal{H} \subseteq \mathcal{T} \times \mathcal{T} \) (resp. \( \mathcal{V} \subseteq \mathcal{T} \times \mathcal{T} \)) represents the horizontal (resp. vertical)
matching relation, and an initial tile type $c \in \mathcal{T}$. A solution for
the instance $(\mathcal{T}, c)$ is a mapping $\tau : [0, t(k, n) - 1] \times
[0, t(k, n) - 1] \rightarrow \mathcal{T}$ such that (first) $\tau(0, 0) = c$, and

**(hor&vert)** for all $i \in [0, t(k, n) - 1]$ and $j \in [0, t(k, n) - 2]$,

\[(\tau(j, i), \tau(j, i + 1), i) \in \mathcal{H} \text{ and } (\tau(i, j), \tau(i, j + 1)) \in \mathcal{V}^t.\]

The problem of checking whether an instance of $\text{Sat}(\mathcal{L}_E)$ has
a solution is known to be $k$-NExpTime-complete (see [36]).

The reduction below from $\text{Sat}(\mathcal{L}_E)$ to $\text{Sat}(\mathcal{L}(\ast))$ recycles
ideas from [7] to reduce $\text{Sat}(\mathcal{L}_E)$ to $\text{Sat}(QK')$. To provide
the adequate adaptation for $\mathcal{L}(\ast)$, we need to solve two major
issues. First, $QK'$ admits second-order quantification, whereas in $\mathcal{L}(\ast)$, the second-order features are limited to
the separating conjunction $\ast$. Second, the second-order quantification of $QK'$ essentially colours the nodes in Kripke-style
structures without changing the frame $(W, R)$. By contrast, the operator $\ast$ modifies the accessibility relation, possibly
making worlds that were reachable from the current world, unreachable in submodels. The Tower-hardness proof for
$\text{Sat}(\mathcal{L}(\ast))$ becomes then much more challenging: we would like to characterise the position on the grid encoded by a
world $w$ by exploiting properties of its descendants (as done for $QK'$), but at the same time, we need to be careful and
only consider submodels where $w$ keeps encoding the same position. In a sense, our encoding is robust: when the operator $\ast$ is used to reason on submodels, we can enforce that no world changes the position of the grid that it encodes.

4.1 Enforcing $t(j, n)$ children.

Let $\mathcal{M} = (W, R, V)$ be a finite forest. We consider two disjoint sets of atomic
propositions $\mathcal{P} = \{p_1, \ldots, p_n, \mathit{val}\}$ and
\[\mathcal{A} = \{x, y, 1, s, r\} \text{ (whose respective role is later defined).}\]

Elements from $\mathcal{A}$ are understood as auxiliary propositions. We call $\mathit{ax}$-node (resp. $\mathit{aux}$-node) a world satisfying the proposition
$\mathit{ax} \in \mathcal{A}$ (resp. satisfying some proposition in $\mathcal{A}$). We call $t$-node a world that satisfies the formula $\mathit{t} \equiv \wedge_{ax \in \mathcal{A}} \neg \mathit{ax}$. Every world of $\mathcal{M}$ is either a $t$-node or an $\mathit{aux}$-node. We say
that $w'$ is a $t$-child of $w \in W$ if $w' \in R(w)$ and $w'$ is a $t$-node. We define the concepts of $\mathit{aux}$-child and $\mathit{aux}$-child similarly.

The key development of our reduction is given by the definition of a formula, of exponential size in $j \geq 1$ and
polynomial size in $n \geq 1$, that when satisfied by $(\mathcal{M}, w)$ forces
every $t$-node in $R^j(w)$, where $0 \leq i < j$, to have exactly $t(j-i, n)$ $t$-children, each of them encoding a different number
in $[0, t(j-i, n) - 1]$. As we impose that $w$ is a $t$-node, it must have $t(j, n)$ $t$-children. We assume $n$ to be fixed throughout
the section and denote this formula by $\mathit{type}(j)$. From the property above, if $\mathcal{M}, w \models \mathit{type}(j)$ then for all $i \in [1, j-1]$ and all $t$-nodes $w' \in R^i(w)$ we have $\mathcal{M}, w' \models \mathit{type}(j-i)$.

First, let us informally describe how numbers are encoded in the model $(\mathcal{M}, w)$ satisfying $\mathit{type}(j)$. Let $i \in [1, j]$. Given
a $t$-node $w' \in R^i(w)$, $\mathit{ni}(w')$ denotes the number encoded by $w'$. We omit the subscript $i$ when it is clear from the context.
When $i = j$, we represent $\mathit{ni}(w')$ by using the truth
values of the atomic propositions $p_1, \ldots, p_n$. The proposition
$p_i$ is responsible for the $b$-th bit of the number, with the least
significant bit being encoded by $p_i$. For example, for $n = 3$, we have $\mathit{ni}(w' \models p_3 \land p_2 \land \neg p_1$ whenever $\mathit{ni}(w') = 6$. The
formula $\mathit{type}(1)$ forces the parent of $w'$ (i.e. is a $t$-node in $R^1(w'))$ to have exactly $2^n$ $t$-children by requiring one $t$-
child for each possible valuation upon $p_1, \ldots, p_n$. Otherwise,
for $i < j$ (and therefore $j \geq 2$), the number $\mathit{ni}(w')$ is represented
by the binary encoding of the truth values of $\mathit{val}$ on the $t$-children of $w'$ which, since $(\mathcal{M}, w') \models \mathit{type}(j-i), are$t(j-i, n)$children implicitly ordered by the number they, in turn, encode. The essential property of $\mathit{type}(j)$ is therefore the following: the numbers encoded by the $t$-children of a $t$-node $w'' \in R^i(w)$, represent positions in the binary representation of the number $\mathit{ni}(w'')$. Thanks to this property, the formula $\mathit{type}(j)$ forces $w$ to have exactly $t(j, n)$ children, all encoding different numbers in $[0, t(j, n) - 1]$. This is roughly represented in the picture below, where “1” stands for $\mathit{val}$
being true whereas “0” stands for $\mathit{val}$ being false.

![Diagram of $t(j, n)$ children](image)

To characterise these trees in $\mathcal{L}(\ast)$, we simulate second-order
quantification by using $\mathit{aux}$-nodes. Informally, we require
a pointed forest $(\mathcal{M}, w)$ satisfying $\mathit{type}(j)$ to be such that (i)
$\forall j$ every $t$-node $w' \in R^j(w)$ has exactly one $x$-child,
and one (different) $y$-child. These nodes do not satisfy any
other auxiliary proposition; (ii) for every $i \geq 2$, every $t$
-node $w'' \in R^i(w)$ has exactly five $\mathit{aux}$-children, one for each
$\mathit{aux}$ in $\mathcal{A}$. We can simulate second-order existential quantification
on $t$-nodes with respect to the symbol $\mathit{ax} \in \mathcal{A}$ by using the operator $\ast$ in order to remove edges leading to
$\mathit{aux}$-nodes. Then, we evaluate whether a property holds on
the resulting model where a $t$-node "satisfies" $\mathit{ax} \in \mathcal{A}$ if it has
a child satisfying $\mathit{ax}$. To better emphasise the need to move along $t$-nodes, given a formula $\phi$, we write $(t)\phi$ for the
formula $\diamond(t \land \phi)$. Dually, $(\exists t)\phi$ is also defined, as expected.

Let us start to formalise this encoding. Let $j \geq 1$. First, we
restrict ourselves to models where every $t$-node reachable in
at most $j$ steps does not have two $\mathit{aux}$-children satisfying
the same proposition. Moreover, these $\mathit{aux}$-nodes have no
children and only satisfy exactly one $\mathit{aux} \in \mathcal{A}$. We express
this condition with the formula $\mathit{in1}(t(j))$ below:

$$\Gamma \wedge \left( \bigwedge_{\mathit{ax} \in \mathcal{A}} (t \Rightarrow \neg (\diamond \mathit{ax} \land \diamond \mathit{ax})) \land \square(\mathit{ax} \Rightarrow \square \bot \land \bigwedge_{bx \in \mathcal{A}} \neg \mathit{bx}) \right).$$
where $\oplus \varphi \triangleq \varphi$ and $\ominus m + \varphi \triangleq \varphi \sqcap \ominus m (\varphi)$. Notice that if $\mathcal{W}, w \models \text{init}(j)$ and $\mathcal{W}' \subseteq \mathcal{W}$, then $\mathcal{W}', w \models \text{init}(j)$.

Among the models $((W', R, V), w)$ satisfying $\text{init}(j)$, we define the ones satisfying $\text{type}(j)$ described below (see similar conditions in [7, Section IV]).

- **(sub)** every $t$-node in $R(w)$ satisfies $\text{type}(j - 1)$;
- **(zero)** there is a $t$-node $w \in R(w)$ such that $\mathcal{W}(w) = 0$;
- **(uniq)** distinct $t$-nodes in $R(w)$ encode different numbers;
- **(compl)** for every $t$-node $w_1 \in R(w)$, if $\mathcal{W}(w_1) < t(j, n) - 1$ then $\mathcal{W}(w_2) = \mathcal{W}(w_1) + 1$ for some $t$-node $w_2 \in R(w)$;
- **(aux)** $w$ is a $t$-node, every $t$-node in $R(w)$ has one $x$-child and one $y$-child, and every $t$-node in $R^2(w)$ has three children satisfying 1, $r$ and $s$, respectively.

We define $\text{type}(0) \triangleq \top$, and for $j \geq 1$, $\text{type}(j)$ is defined as

$\text{type}(j) \triangleq \text{sub}(j) \land \text{zero}(j) \land \text{uniq}(j) \land \text{compl}(j) \land \text{aux}$,

where each conjunct expresses its homonymous property.

The formulae for $\text{sub}(j)$, $\text{aux}$ and $\text{zero}(j)$ can be defined as

$\text{sub}(j) \triangleq [t] \text{type}(j - 1)$;
$
\text{aux} \triangleq t \land \{t\} \land \varphi(y) \land [t]^2(\varphi \land s \land \varphi);$
$
\text{zero}(j) \triangleq t \land \varphi(y) \land [t]^2(\varphi \land s \land \varphi);$
$
\text{zero}(j + 1) \triangleq \{t\} \land \varphi(y).$

The challenge is therefore how to express $\text{uniq}(j)$ and $\text{compl}(j)$, to guarantee that the numbers of children of $w$ span all over $[0, [t, j, n] - 1]$. The structural properties expressed by $\text{type}(j)$ lead to strong constraints, which permits to control the effects of $*$ when submodels are constructed. This is a key point in designing $\text{type}(j)$ as it helps us to control which edges are lost when considering a submodel.

### Nominals, forks and number comparisons

In order to define $\text{uniq}(j)$ and $\text{compl}(j)$ (completing the definition of $\text{type}(j)$), we introduce auxiliary formulae, characterising classes of models that emerge naturally when trying to capture the semantics of $\text{uniq}(j)$ and $\text{compl}(j)$.

Let us consider a finite forest $\mathcal{W} = (W, R, V)$ and $w \in W$. A first ingredient is given by the concept of local nominals, borrowed from [7]. We say that $ax \in \text{Aux}$ is a (local) nominal for the depth $i \geq 1$ if there is exactly one $t$-node $w' \in R^t(w)$ having an $ax$-child. In this case, $w'$ is said to be the world that corresponds to the local nominal $ax$. The following formula states that $ax$ is a local nominal for the depth $i$:

$\text{nom}_i(ax) \triangleq (t)^i \land \varphi(ax) \land \bigwedge_{k \in [0, i - 1]} [t]^k \land \varphi(ax \land \varphi)$.

We define the formula $\ominus ax \varphi \triangleq (t)^i \land \varphi(ax \land \varphi)$, which, under the hypothesis that $\text{ax}$ is a local nominal for the depth $i$, states that $\varphi$ holds on the $t$-node that corresponds to $ax$. Moreover, we define $\ominus ax \varphi \land \varphi \triangleq \text{nom}_i(ax) \land \text{nom}_i(bx) \land \lnot \ominus ax \land \varphi$, which states that $ax$ and $bx$ are two nominals for the depth $i$ with respect to two distinct $t$-nodes.

As a second ingredient, we introduce the notion of fork that is a specific type of models naturally emerging when trying to compare the numbers $\mathcal{W}(w_1)$ and $\mathcal{W}(w_2)$ of two worlds $w_1, w_2 \in R^t(w)$ (e.g. when checking whether $\mathcal{W}(w_1) = \mathcal{W}(w_2)$ or $\mathcal{W}(w_1) = \mathcal{W}(w_2) + 1$ holds). Given $j \geq i \geq 1$ we introduce the formula $\text{fork}_j(ax, bx)$ that is satisfied by $(\mathcal{W}, w)$ iff:

- $ax$ and $bx$ are nominals for the depth $i$.
- $w$ has exactly two $t$-children, say $w_1$ and $w_2$.
- For every $k \in [1, i - 1]$, both $R^k(w_1)$ and $R^k(w_2)$ contain exactly one $t$-child.
- The only $t$-node in $R^k\{w_j\}$, say $w_{ax}$, corresponds to the nominal $ax$. The only $t$-node in $R^k\{w_j\}$, say $w_{bx}$, corresponds to the nominal $bx$.
- If $i < j$, then $(\mathcal{W}, w_{ax})$ and $(\mathcal{W}, w_{bx})$ satisfy $\text{type}_{\text{sr}}(j - i) \triangleq \text{type}(j - i) \land \{t\} \land s \land \varphi$.

It should be noted that, whenever $(\mathcal{W}, w)$ satisfies the formula $\text{fork}_j(ax, bx)$, we witness two paths of length $i$, both starting at $w$ and leading to $w_{ax}$ and $w_{bx}$, respectively. Worlds in this path may have Aux-children. Below, we schematise a model satisfying $\text{fork}_j(ax, bx)$:

![Model satisfying fork](image)

Since the definition of $\text{fork}_j(ax, bx)$ is recursive on $i$ and $j$ (due to $\text{type}(j - i)$), we postpone its formal definition to the next two sections where we treat the base cases for $i = j$ and the inductive case for $j > i$ separately.

The last auxiliary formulae are $[ax < bx]^j$ and $[bx = ax + 1]^j$. Under the hypothesis that $(\mathcal{W}, w)$ satisfies $\text{fork}_j(ax, bx)$, the formula $[ax < bx]^j$ is satisfied whenever the two (distinct) worlds $w_{ax}$, $w_{bx} \in R^t(w)$ corresponding to the nominals $ax$ and $bx$ are such that $\mathcal{W}(w_{ax}) < \mathcal{W}(w_{bx})$. Similarly, under the hypothesis that $(\mathcal{W}, w)$ satisfies $\text{fork}_j(ax, bx)$, the formula $[bx = ax + 1]^j$ is satisfied whenever $\mathcal{W}(w_{bx}) = \mathcal{W}(w_{ax}) + 1$ holds. Both formulae are recursively defined, with base cases for $i = j$ and $j = 1$, respectively.

For the base case, we define the formulae $\text{fork}_j(ax, bx)$ and $[ax < bx]^j$ (for arbitrary $j$), as well as $[bx = ax + 1]^j$. From these formulae, we are then able to define $\text{uniq}(i)$ and $\text{compl}(i)$, which completes the characterisation of $\text{type}(1)$ and $\text{type}_{\text{sr}}(1)$. Afterwards, we consider the case $1 \leq i < j$ and $j \geq 2$, and define $\text{fork}_{j-1}(ax, bx)$, $[ax < bx]^j$, $[bx = ax + 1]^j$, as well as $\text{uniq}(j)$ and $\text{compl}(j)$, by only relying on formulae that are already defined (by inductive reasoning).

### Base cases: $i = j$ or $j = 1$

In what follows, we consider a finite forest $\mathcal{W} = (W, R, V)$ and a world $w$. Following its informal description, we have

$\text{fork}_j(ax, bx) \triangleq \{z \neq t \land [t] \ominus z t \land \varphi(ax) \land \varphi(bx) \land \lnot \ominus ax \land \varphi$, where $\ominus ax \varphi \triangleq \top$ for $j < 0$. As previously explained, in the base case, the number $\mathcal{W}(w')$ encoded by a $t$-node $w' \in R^t(w)$
is represented by the truth values of $p_1, \ldots, p_n$. Then, the formula $[ax < bx]_j$ is defined as

$$[ax < bx]_j \equiv \bigvee_{u \in [1, n]} (\neg p_u \land \bigvee_{v \in [1, u-1]} (\neg p_v \land ((\neg p_u \land p_v) \leftrightarrow \bigvee_{v \in [1, u+1]} (\neg p_v \land p_u)))$$

The satisfaction of $(\mathcal{W}, w) \models \text{fork}_i^j(ax, bx)$ enforces that the distinct t-nodes $w_{ax}, w_{bx} \in R'(w)$ corresponding to $ax$ and $bx$ satisfy $n(w_{ax}) < n(w_{bx})$, which can be shown by using standard properties about bit vectors.

The formula $[bx = ax+1]_j$ is similarly defined:

$$\bigvee_{u \in [1, n]} (\neg p_u \land \bigvee_{v \in [1, u-1]} (\neg p_v \land \bigvee_{v \in [1, u+1]} (\neg p_v \land p_u)))$$

Assuming $(\mathcal{W}, w) \models \text{fork}_i^j(ax, bx)$, this formula states that the two distinct t-nodes $w_{ax}, w_{bx} \in R'(w)$ corresponding to $ax$ and $bx$ are such that $n(w_{ax}) = n(w_{bx}) + 1$. Again, correctness is guaranteed by standard analysis on bit vectors.

To define $\text{uniq}(1)$, we recall that a model satisfying $\text{type}(1)$ satisfies the formula aux and hence every t-node in $R(w)$ has two auxiliary children, one $x$-node and one $y$-node. The idea is to use these two Aux-children and rely on $\ast$ to state that it is not possible to find a submodel of $\mathcal{W}$ such that $w$ has only two distinct children $w_x$ and $w_y$ and $bx$ simulates a second-order quantifier on $x$ and $y$. Let $[x = y]_i \equiv \neg(x < y) \lor (y < x)$. We define $\text{uniq}(1) \equiv \neg(\ast \circ (\text{fork}_1^i(x, y) \land [x = y]_i))$.

To capture $\text{compl}(1)$ we state that it is not possible to find a submodel of $\mathcal{W}$ that looses $x$-nodes from $R'(w)$, keeps all y-nodes, and is such that (i) $x$ is a local nominal for the depth 1, corresponding to a world $w_x$ encoding $n(w_{ax}) < 2^n - 1$; (ii) there is no submodel where $w$ has two t-children, $w_x$ and a second world $w_y$, such that $w_x$ corresponds to the nominal $y$ and $n(w_y) = n(w_{bx}) + 1$. Thus, $\text{compl}(1)$ is defined as:

$$\neg(\sqcap_{x \in [1, n]} P_t \land (\sqcup_{x \in [1, n]} P_t \land \neg (\ast \circ (\text{fork}_1^i(x, y) \land [y = x+1]))))$$

The subscript “$1$” in the formula 1 refers to the fact that we are treating the base case of $\text{compl}(j)$ with $j = 1$. We have $1 \equiv \exists_{i \in [1, n]} P_t$, reflecting the encoding of $2^n - 1$.

This concludes the definition of $\text{type}(1)$ and $\text{type}_{\text{sr}(1)}$, which is established correct with respect to its specification.

**Lemma 4.1.** Let $\mathcal{W}, w \models \text{init}(1)$. We have $\mathcal{W}, w \models \text{type}(1) \iff (\mathcal{W}, w)$ satisfies (sub1), (zero1), (uniq1), (compl1) and (aux).

**Inductive case:** $1 \leq i < j$. As an implicit inductive hypothesis used to prove that the formulae are well-defined, we assume that $[bx = ax+1]_j$ and $\text{type}(j')$ are already defined for every $j' < j$, whereas $\text{fork}_i^j(ax, bx)$, and $[ax < bx]_j$ are already defined for every $1 \leq i' \leq j'$ such that $j' - i' < j - i$. Therefore, we define:

$$\text{fork}_i^j(ax, bx) \equiv \text{fork}_i^j(ax, bx) \land [i']^j \text{type}_{\text{sr}(1)}(j - i).$$

It is easy to see that this formula is well-defined: $\text{fork}_i^j(ax, bx)$ is from the base case, whereas $\text{type}_{\text{sr}(1)}(j - i)$ is defined by inductive hypothesis, since we have $j - i < j$.

Consider now $[ax < bx]_j$. Assuming $\mathcal{W}, w \models \text{fork}_j^j(ax, bx)$, we wish to express $n(w_{ax}) < n(w_{bx})$ for the two distinct worlds $w_{ax}, w_{bx} \in R'(w)$ corresponding to the nominals $ax$ and $bx$, respectively. As $i < j$, $n(w_{ax})$ (resp. $n(w_{bx})$) is encoded using the truth value of $\text{val}$ on the $t$-children of $w_{ax}$ (resp. $w_{bx}$). To rely on arithmetical properties of binary numbers used to define $[ax < bx]_j$, we need to find two partitions $P_{ax} = \{L_{ax}, S_{ax}, R_{ax}\}$ and $P_{bx} = \{L_{bx}, S_{bx}, R_{bx}\}$, one for the $t$-children of $w_{ax}$ and another one for those of $w_{bx}$ s.t.

**LSR:** Given $b \in \{ax, bx\}$, $P_b$ splits the $t$-children as follows:

- there is a $t$-child $S_b$ of $w_b$ such that $S_b = \{s_b\};$
- $n(i) < n(s_b) < n(l)$, for every $r \in R_b$ and $l \in L_b.$

**LESS:** $P_{ax}$ and $P_{bx}$ have constraints to satisfy $<$:

- $n(s_{ax}) = n(s_{bx}), \mathcal{W}, s_{ax} \models \neg \text{val}$ and $\mathcal{W}, s_{ax} \models \text{val};$
- for every $l_{ax} \in L_{ax}$ and $l_{bx} \in L_{bx}$, if $n(l_{ax}) = n(l_{bx})$ then $\mathcal{W}, l_{ax} \models \text{val}$ iff $\mathcal{W}, l_{bx} \models \text{val}.$

It is important to notice that these conditions essentially revolve around the numbers encoded by $t$-children, which will be compared using the already defined (by inductive reasoning) formula $[ax < bx]_j$, where $j' - i' < j - i$. Since the semantics of $[ax < bx]_j$ is given under the hypothesis that $\mathcal{W}, w \models \text{fork}_j^j(ax, bx)$, we can assume that every child of $w_{ax}$ and $w_{bx}$ has all the possible $\text{Aux}$-children. Then, we rely on the auxiliary propositions in $\{1, s, r\}$ in order to mimic the reasoning done in (LSR) and (LESS).

We start by considering the constraints involved in (LSR) and express them with the formula $1\text{sr}(j)$, which is satisfied by a pointed forest $(\mathcal{W} = (W, R, V), v)$ whenever:

- $(\mathcal{W}, w)$ satisfies type$(j)$.
- Every $t$-child of $w$ has exactly one $\{1, s, r\}$-child, and only one of these $t$-children (say $w'$) has an $s$-child.
- Every $t$-child of $w$ that has an $1$-child (resp. $r$-child) encodes a number greater (resp. smaller) than $n(w')$.

Despite this formula being defined in terms of $\text{type}(j)$, we only rely on $1\text{sr}(j - i)$ (which is defined by inductive reasoning) in order to define $[ax < bx]_j$. The picture below schematises a model satisfying $1\text{sr}(j)$.
where $S^j_i(ax, bx)$ and $L^j_i(ax, bx)$ check the first and second condition in (LESS), respectively. In particular, by defining $[ax = bx]^j_{i \oplus} = ((ax < bx)^j_i \lor (bx < ax)^j_i)$, we have

$$S^j_i(ax, bx) \equiv \top \lor (fork^j_i(x, y) \land \neg \left( \begin{array}{l} \neg \left( \begin{array}{l} ax \neq bx \land bx \neq ax \land x \neq y^j_i \land \neg \left( \begin{array}{l} x^j_i = val \lor y^j_i = val \end{array} \right) \right) \right) \right)$$

and

$$L^j_i(ax, bx) \equiv \neg \left( \begin{array}{l} \top \lor (fork^j_i(x, y) \land \neg \left( \begin{array}{l} ax \neq bx \land bx \neq ax \land x \neq y^j_i \land \neg \left( \begin{array}{l} x^j_i = val \lor y^j_i = val \end{array} \right) \right) \right) \right)$$

Both $fork^j_i(x, y)$ and $[x = y]^j_i$ are used in these formulae are defined recursively. The formula $S^j_i(ax, bx)$ states that there is a submodel $M \subseteq \mathcal{M}$ such that

1. \[ \mathcal{M}', \omega \models fork^j_i(x, y) \]
2. $s_{ax}$ corresponds to the nominal $x$ at depth $i + 1$.
3. $s_{bx}$ corresponds to the nominal $y$ at depth $i + 1$.
4. $\forall \omega \in \mathcal{M}, s_{ax} \neq val \land s_{bx} \models val$.

(The enumeration I-VI refers to the conjuncts in the formula $S^j_i(ax, bx)$ correctly models the first condition of (LESS).

Regarding $L^j_i(ax, bx)$ and (LESS), a similar analysis can be performed. We define $LS_i^j(ax, bx) \equiv L_i^j(ax, bx) \land S_i^j(ax, bx)$.

Let us consider $[bx = ax+1]$. Under the hypothesis that $\mathcal{M}, \omega \models fork^j_i(ax, bx)$, this formula must express $\mathcal{M}([w_{ax}]) = \mathcal{M}([w_{bx}]) + 1$ for the two (distinct) worlds $w_{ax}, w_{bx} \in R^i(\omega)$. Then, as done for defining $[ax < bx]^j_i$, we take advantage of arithmetical properties on binary numbers and we search for two partitions $P_{ax} = \{L_{ax}, S_{ax}, R_{ax}\}$ and $P_{bx} = \{L_{bx}, S_{bx}, R_{bx}\}$ of the $t$-children of $w_{ax}$ and $w_{bx}$, respectively, such that $P_{ax}$ and $P_{bx}$ satisfy (LSR) as well as the condition below:

(PLUS): $P_{ax}$ and $P_{bx}$ have the arithmetical properties of $+$:

1. For every $r_{ax} \in R_{ax}$, we have $\mathcal{M}, r_{ax} \models val$.
2. For every $r_{bx} \in R_{bx}$, we have $\mathcal{M}, r_{ax} \neq val$.

where $S_{ax} = \{S_{ax}\}$ and $S_{bx} = \{S_{bx}\}$, as required by (LSR).

The definition of $[bx = ax+1]$ is similar to $[ax < bx]^j_i$:

$$\top \lor (\neg (nom_1(ax \neq bx) \lor [t] \neg sr(j-1) \land LS_j(ax, bx) \land R(ax, bx))$$

where $R(ax, bx) \equiv \left( \begin{array}{l} \neg \left( \begin{array}{l} ax \neq bx \land ax \neq bx \land x \neq y^j_i \land \neg \left( \begin{array}{l} x^j_i = val \lor y^j_i = val \end{array} \right) \right) \right) \right)$ and $\neg (nom_1(ax \neq bx) \lor [t] \neg sr(j-1) \land LS_j(ax, bx) \land R(ax, bx))$ captures the last two conditions of (PLUS).

To define $uniq(j)$ and $comp(j)$, we rely on $fork^j_i(ax, bx)$, $[ax < bx]^j_i$ and $[bx = ax+1]^j_i$.

$$\text{uniq}(j) \equiv \neg \left( \top \lor (fork^j_i(x, y) \land [x = y]^j_i) \right)$$

$$\text{comp}(j) \equiv \neg \left( \top \lor (fork^j_i(x, y) \land [x = y+1]^j_i) \right)$$

where $1_i \equiv [t]val$ reflects the encoding of $(t, n - 1)$ for $j > 1$. The main difference between $comp(1)$ and $comp(j)$ ($j > 1$) is that the conjunct $[t] \neg y$ of $comp(1)$ is replaced by $[t] \neg y$ in $comp(j)$, as needed to correctly evaluate $fork^j_i(x, y)$. Indeed, the difference between $fork^j_i(x, y)$ and $fork^j_i(x, y)$ is precisely that the latter requires $[t]type_{LS}(j-1)$. The definition of $type(j)$ is now complete. We can state its correctness.

**Lemma 4.2.** Let $\mathcal{M}, \omega \models init(j)$. We have $\mathcal{M}, \omega \models type(j)$ iff $(\mathcal{M}, \omega)$ satisfies (sub), (zero), (uniq), (compl) and (aux).

The size of $type(j)$ is exponential in $j > 1$ and polynomial in $n \geq 1$. As its size is elementary, we can use this formula as a starting point to reduce $Tile_k$.

### 4.2 Tiling a grid $[0, t(k, n) - 1] \times [0, t(k, n) - 1]$

Below, we briefly explain how to use previous developments to define a uniform reduction from $Tile_k$, for every $k \geq 2$.

Several adaptations are needed to encode smoothly the grid but the hardest part was the design of $type(j)$. Let $k \geq 2$ and $(\mathcal{T}, c)$ be an instance of $Tile_k$. We can construct a formula $tiling_{\mathcal{T}, c}(k)$ that is satisfiable if and only if $(\mathcal{T}, c)$ as a solution. To represent $[0, t(k, n) - 1]^2$ in some pointed forest $(\mathcal{M}, \omega)$, where $\mathcal{M} = (W, R, V)$, we recycle the ideas for defining $type(k)$. From Lemma 4.2, we know that if $\mathcal{M}, \omega \models init(k) \land type(k)$ then the $t$-children of $\omega$ encode the interval $[0, t(k, n) - 1]$. A position in the grid is however a pair of numbers, hence the *crux of our encoding* rests on the fact that each $w' \in R(w)$ encodes two numbers $\mathcal{M}_{\mathcal{T}}(w')$ and $\mathcal{M}_{\mathcal{T}}(w')$. Similarly to $type(k)$, these numbers are represented by the truth values on the $t$-children of $w'$, with the help of new propositions $\mathcal{M}_{\mathcal{T}}(val_{H})$ and $\mathcal{M}_{\mathcal{T}}(val_{V})$. We are in luck: since both numbers are from $[0, t(k, n) - 1]$, $w'$ just needs as many children as when encoding a single number, and therefore if $\mathcal{M}, \omega \models tiling_{\mathcal{T}, c}(k)$ then $\mathcal{M}, \omega \models type(k-1)$. In fact, the portion of $tiling_{\mathcal{T}, c}(k)$ that encodes the grid can be described quite naturally by slightly updating the characterisation of $type(k)$. For example, $\text{(uniq)}$ becomes $\text{(uniq)}_{\mathcal{T}, k}$ for all distinct $t$-nodes $w_1, w_2 \in R(w)$ $\mathcal{M}_{\mathcal{T}}(w_1) \neq \mathcal{M}_{\mathcal{T}}(w_2)$ or $\mathcal{M}_{\mathcal{T}}(w_1) \neq \mathcal{M}_{\mathcal{T}}(w_2)$.

The formula $\text{uniq}_{\mathcal{T}, k}(h)$ has to be updated accordingly, but without major differences or complications. Of course, more is required as $tiling_{\mathcal{T}, c}(k)$ must also encode the tiling conditions (first) and (hor&vert). Fortunately, the kit of formulae defined for $type(k)$ allows us to have access to $\mathcal{M}_{\mathcal{T}}(w')$ and $\mathcal{M}_{\mathcal{T}}(w')$ in such a way that both conditions can be expressed rather easily. For example, to express vertical constraints, we design a formula stating that for all $t$-nodes $w_1, w_2 \in R(w)$, if $\mathcal{M}_{\mathcal{T}}(w_2) = \mathcal{M}_{\mathcal{T}}(w_1) + 1$ and $\mathcal{M}_{\mathcal{T}}(w_2) = \mathcal{M}_{\mathcal{T}}(w_1)$ then there is $(c_1, c_2) \in V$ such that $w_1 \in V(c_1)$ and $w_2 \in V(c_2)$. Further details are omitted by lack of space.

**Theorem 4.3.** Sat(ML(\ast)) is Tower-complete.

### 5 ML(\ast) Strictly Less Expressive Than GML

Below, we focus on the expressivity of ML(\ast). We first show $ML(\ast) \subseteq GML$ and then we prove the strictness of the inclusion. The former result takes advantage of the notion of g-bisimulation, i.e. the underlying structural indistinguishability relation of GML, studied in [20]. To show $ML(\ast) \subseteq GML$, we define an ad hoc notion of Ehrenfeucht-Fraissé games for $ML(\ast)$, see e.g. classical definitions in [31] and similar.
approaches in [13, 18]. Then, we design a simple formula in GML that cannot be expressed in ML(*).

5.1 ML(*) is not more expressive than GML
To establish that ML(*) ≤ GML, we proceed as in Section 3.1. In fact, by Lemma 2.2, given ϕ1, ϕ2 in GML, the formula ϕ1 ∨ ϕ2 is equivalent to φ[ϕ1, ϕ2]. Moreover, we know that given ϕ1, ϕ2 in GML, φ[ϕ1, ϕ2] is equivalent to some formula in GML, as shown in Section 3. So, to prove that ML(*) ≤ GML by applying the proof schema of Theorem 3.2, it is sufficient to show that given ϕ in GML, there is ψ in GML such that φ ≡ ψ. To do so, we rely on the indistinguishability relation of GML, called g-bisimulation [20].

A g-bisimulation is a refinement of the classical back-and-forth conditions of a bisimulation (see e.g. [8]), tailored towards capturing graded modalities. It relates models with similar structural properties, but up to parameters m, k ∈ N responsible for the modal degree and the graded rank, respectively. The following invariance result holds: g-bisimilar models are modally equivalent and-forth conditions of a bisimulation (see e.g. [8]), tailored by applying the proof schema of Theorem 3.2, it is sufficient

Lemma 5.1. Let (M, w) ⊨ P m,k \((\psi', w')\) where m, k ∈ N, P ⊨ AP, M = (W, R, V) and M' = (W', R', V'). Let R₁ ⊆ R. There is R₁' ⊆ R' s.t. ((W, R₁, V), w) ≡ (w') \((\psi', R', V), w')\) and if R₁(w) = R(w), then R₁'(w') = R'(w').

The proof of Lemma 5.1 is by induction on m. The last condition about R₁(w) = R(w) will serve in the proof of Lemma 5.2, as it allows us to capture the semantics of ◗, by preserving the children of the world \(w\). In the proof, we rely on the properties of g-bisimulations [20] to define a binary relation ↔ between worlds of R(w) and R'(w'). Every w₁ ↔ w₁' is such that (M, w₁) \(\equiv\) (M', w₁'). The operator ◗ does not necessarily preserve the children of w₁ and w₁', so that the induction hypothesis, naturally defined from the statement of Lemma 5.1, is applied on models where the condition R₁(w₁) = R(w₁) may not hold. We show that for all R₁ ⊆ R, it is possible to construct R₁' ⊆ R' such that, for all w₁ ↔ w₁', ((W, R₁, V), w₁) \(\equiv\) (W', R', V'), w₁'). The result is then lifted to ((W, R₁, V), w) ≡ (W', R', V'), w), again thanks to the properties of the g-bisimulation.

Intuitively, Lemma 5.1 states that given two models satisfying the same formulae up to the parameters m and (m, k), we can extract submodels satisfying the same formulae up to m and k (reduced graded rank). This allows us to conclude that if ϕ is in GML, there is some GML formula equivalent to ◗(ϕ) (Lemma 5.2). In other words, the operator ◗ can be eliminated to obtain a GML formula. This, together with Lemma 2.2 and Theorem 3.2 entail ML(*) ≤ GML.

Lemma 5.2. For every ϕ ∈ GML[m, k, P] there is a formula ψ ∈ GML[m, f(m, k), P] such that ◗(ϕ) ≡ ψ.

5.2 Showing ML(*) < GML with EF games for ML(*)
We tackle the problem of showing that ML(*) is strictly less expressive than GML. To do so, we adapt the notion of Ehrenfeucht-Fraïssé games (EF games, in short) [31] to ML(*), and use it to design a GML formula that is not expressible in ML(*). We write ML(*)[m, s, P] for the set of formulae \(\varphi\) of ML(*) having md(\(\varphi\)) ≤ m, at most s nested *, and atomic propositions from P ⊨ AP. It is easy to see that ML(*)[m, s, P] is finite up to logical equivalence.

We introduce the EF games for ML(*). A game is played between two players: the spoiler and the duplicator. A game state is a triple made of two pointed forests (M, w) and (M', w') and a rank (m, s), where m, s ∈ N and P ⊨ AP. The goal of the spoiler is to show that the two models are different. The goal of the duplicator is to counter the spoiler and to show that the two models are similar. Two models are different whenever there is \(\varphi\) in ML(*)[m, s, P] that is satisfied by only one of the two models. The EF games for ML(*) are formally defined in Figure 1. The exact correspondence between the game and the logic is formalised in Lemma 5.3.

Using the standard definitions in [31], the duplicator has a winning strategy for the game ((M, w), (M', w'), (m, s, P))
Game on $[(\mathcal{W}_1, \{W_1, R_1, V_1\}), \mathcal{W}_2, \mathcal{W}_3, (m, s, P)]$. If there is $p \in P$ s.t. $W_1 \in V(p)$ iff $W_2 \notin V(p)$ then the spoiler wins. Else, the spoiler chooses $i \in \{1, 2\}$ and plays on $\mathcal{W}_i$. The replicator duplicates on $\mathcal{W}_j$ where $j \neq i$. The spoiler must choose one of the following moves, otherwise the duplicator wins:

- **modal move**: if $m \geq 1$ and $R_1(w_i) \neq \emptyset$ then the spoiler can choose to play a modal move by selecting an element $w_i' \in R_1(w_i)$. Then,
  - the replicator must reply with a $w_i' \in R_i(w_i)$ (else, the spoiler wins);
  - the game continues on $[(\mathcal{W}_1, w_i'), (\mathcal{W}_2, w_i'), (m - 1, s, P)]$.

- **spatial move**: if $s \geq 1$ then the spoiler can choose to play a spatial move by selecting two finite forests $\mathcal{W}_1$ and $\mathcal{W}_2$ s.t. $R_1(w_i) + R_2(w_i) = \mathcal{W}_i$. Then,
  - the replicator duplicates with two forests $\mathcal{W}_1'$ and $\mathcal{W}_2'$ s.t. $\mathcal{W}_1' + \mathcal{W}_2' = \mathcal{W}_i$;
  - the game continues on $[(\mathcal{W}_1', w_i), (\mathcal{W}_2', w_i), (m, s - 1, P)]$, where $k \in \{1, 2\}$ is chosen by the spoiler.

**Figure 1.** Ehrenfeucht-Fraïssé games for ML($\ast$).

If she can play in a way that guarantees her to win regardless of how the spoiler plays. When this is the case, we write $(\mathcal{W}, w) \models_{m,s}^{P} (\mathcal{W}', w')$. Similarly, the spoiler has a winning strategy, written $(\mathcal{W}, w) \not\models_{m,s}^{P} (\mathcal{W}', w')$, if he can play in a way that guarantees him to win, regardless of how the duplicator plays. Lemma 5.3 guarantees that the games are well-defined.

**Lemma 5.3.** $(\mathcal{W}, w) \not\models_{m,s}^{P} (\mathcal{W}', w')$ iff there is a formula $\varphi$ in ML($\ast$) such that $(\mathcal{W}, w) \models \varphi$ and $(\mathcal{W}', w') \nvDash \varphi$.

Lemma 5.3 is proven with standard arguments from [31], for instance the left-to-right direction, i.e. the completeness of the game, is by induction on the rank $(m, s, P)$. Thanks to the EF games, we are able to find a GML formula $\varphi$ that is not expressible in ML($\ast$). By Lemma 2.1 and as ML($\mathcal{J}$) $\equiv$ GML, such a formula is necessarily of modal degree at least 2. Happily, $\varphi = \square_{n \geq 2} \triangledown_{s = 1} \top$ does the job and cannot be expressed in ML($\ast$). For the proof, we show that for every rank $(m, s, P)$, there are two structures $(\mathcal{W}, w)$ and $(\mathcal{W}', w')$ such that $(\mathcal{W}, w) \not\models_{m,s}^{P} (\mathcal{W}', w')$, $(\mathcal{W}, w) \models \varphi$ and $(\mathcal{W}', w') \nvDash \varphi$. The inexpressibility of $\varphi$ then stems from Lemma 5.3. The two structures are represented below ($(\mathcal{W}, w)$ on the left).

In the following, we say that a world has **type $i$** if it has $i$ children. As one can see in the figure above, children of the current worlds $w$ and $w'$ are of three types: 0, 1 or 2. When the spoiler performs a spatial move in the game, a world of type $i$ can take, in the submodels, a type between 0 and $i$. That is, the number of children of a world weakly monotonically decreases when taking submodels. This monotonicity, together with the finiteness of the game, lead to bounds on the number of children of each type, over which the duplicator is guaranteed to win. For instance, the bound for worlds of type 2 is given by the value $2^2(s+1)(s+2)+1$, where $s$ is the number of spatial moves in the game. In the two presented pointed forests, one child of type 0 and one of type 2 are added with respect to these bounds, so that the duplicator can make up for the different numbers of children of type 1.

**Lemma 5.4.** ML($\ast$) cannot characterise the class of models satisfying the GML formula $\square_{n \geq 2} \triangledown_{s = 1} \top$.

Notice that ML($\ast$) is more expressive than ML. Indeed, the formula $\square \top \otimes \square \top$ distinguishes the two models on the right, which are bisimilar and hence indistinguishable in ML [46]. By ML($\ast$) $\subseteq$ GML, Lemma 5.4 and Theorem 3.2, we conclude.

**Theorem 5.5.** ML $<$ ML($\ast$) $<$ GML $\cong$ ML($\mathcal{J}$).

Below, we show how our new results on ML($\mathcal{J}$) and ML($\ast$) allow us to make substantial contributions for sister logics.

6 ML($\mathcal{J}$), ML($\ast$) and Sister Logics

Static ambient logic (SAL) is a formalism proposed to reason about spatial properties of concurrent processes specified in the ambient calculus [15]. In [12], the satisfiability and validity problems for a very expressive fragment of SAL are shown to be decidable and conjectured to be in PSPACE (see [12, Section 6]). We invalidate this conjecture by showing that the intensional fragment of SAL (see [52]), herein denoted SAL($\mathcal{J}$), is already AExp$_{Pol}$-complete. More precisely, we design semantically faithful reductions between Sat(SAL($\mathcal{J}$)) and Sat(SA($\mathcal{J}$)) (in both directions), leading to the above-mentioned result by Corollary 3.8. SAL($\mathcal{J}$) formulae are from $\varphi = \top | \emptyset | n[\varphi] | \varphi \land \varphi | \neg \varphi | \varphi[n]$, where $n \in AP$ is an ambient name. Historically, the semantics of SAL is given on a class of syntactically defined finite trees. However, this class of models is isomorphic to the class of finite trees $W = (W, R, V)$, such that each world in $W$ satisfies exactly one atomic proposition (its ambient name). Then, the satisfaction relation $|=\mathcal{J}$ for SAL($\mathcal{J}$) is standard for $\top$ and Boolean connectives, $\varphi^n_{\mathcal{J}}$ is as in ML($\mathcal{J}$), and otherwise:

- $(\mathcal{W}, w) |= \emptyset \Longleftrightarrow R(w) = \emptyset$;
- $(\mathcal{W}, w) |= n[\varphi] \Longleftrightarrow$ there is $w' \in W$ such that $R(w) = \{w'\}$, $w' \in V(n)$ and $w' \models \varphi$.

With such a presentation, SAL($\mathcal{J}$) is a fragment of ML($\mathcal{J}$), where $\emptyset$ and $n[\varphi]$ correspond to $\square \bot$ and $\square_{n \geq 1} \top \land \square \land (n \land \varphi)$, respectively. However, to reduce Sat(SAL($\mathcal{J}$)) to Sat(MA($\mathcal{J}$)), we must deal with the constraint on $V$ (uniqueness of the ambient name). Let $\varphi$ be in SAL($\mathcal{J}$) written with the ambient names in $N = \{n_1, \ldots, n_j\}$. It is known (see [12, Lemma 8]) that if $\varphi$ is satisfiable, then it can be satisfied by a tree having ambient names from $N \cup \{\overline{n}\}$, where $\overline{n}$ is a fresh name. Thus, we can show that $\varphi$ is satisfiable iff so is the ML($\mathcal{J}$) formula $\varphi \land \square_{n \neq m}[\varphi](\bigwedge_{n \in N \cup \overline{n}}(n \land \bigwedge_{i \in \overline{n}} n_i \land (n_i \land \neg n_i)))$, where the right conjunct states that $V$, restricted to the propositions in $N \cup \{\overline{n}\}$, forms a partition of the worlds reachable from the current one in at most md($\varphi$) steps.
Reducing Sat(ML()) to Sat(SAL()) requires a bit more work. Let \( W = (W, R, V) \) be a finite forest and \( w \in W \). Assume we want to check the satisfiability status of \( \varphi \) in ML() having atomic propositions from \( P = \{p_1, \ldots, p_m\} \) with \( n \) occurrences of \( \tau \). We encode \( (\mathfrak{M}, w) \) into a model \( (\mathfrak{M}' = (W', R', V'), w) \) of SAL() as follows. Let \( r \) and \( ap \) be two ambient names not in \( P \). The ambient name \( r \) encodes the relation \( R \) whereas \( ap \) can be seen as a container for propositional variables holding the current world. (i) We require \( W \subseteq W' \), \( R' \subseteq R' \) and \( \bigcup_{i \in \{1, \ldots, \text{md}(\varphi)\}} R'(w) \subseteq V'(\text{rel}) \), i.e., every world reachable from \( w \) in at most \( \text{md}(\varphi) \) steps has the ambient name \( r \). Let \( w' \) be one of these worlds and suppose that \( \{\varphi | w' \in V(\varphi) \cap P = \{q_1, \ldots, q_i\} \). (ii) We require \( W' \) to contain \( n + 1 \) worlds \( w'_1, \ldots, w'_{n+1} \in R'(w') \) all having ambient name \( ap \). These worlds encode copies of \( w' \)'s valuation, similarly to what is done in Section 3.2 to encode teams from PL[-]. (iii) For all \( j \in [1, n+1] \), \( R'(w'_j) \) contains \( l \) worlds, all satisfying \( \varphi \) and a distinct ambient name from \( \{q_1, \ldots, q_l\} \). Below we schematise the encoding (w.r.t. \( w' \)).

Let \( n \in AP \). We define the modality \( (n)\varphi \equiv n[\varphi] \) and its dual \( [n]\varphi \equiv \neg(n)\neg\varphi \). We write \( V[n] \) for \( \forall x. (\neg \varphi \land n[x] \neg\varphi) \), so that \( (\mathfrak{M}, w) \models V[n] \) whenever every child of \( w \) has the ambient name \( n \). Moreover, \( \# \geq 0 \models \top \) and \( \# \geq \beta + 1 \models \neg\varphi \neg\beta \) whenever \( w \) has at least \( \beta \) children. Lastly, \( \# = \beta \models \# \land \neg\# \geq \beta + 1 \). The models of SAL() encoding models of ML() are characterised by

\[
C_{\varphi} = \bigwedge_{j \in \{0, \text{md}(\varphi)\}} \left( r \in V[ap] \land [\# = n+1] \land [ap](p_1[0] \neg\varphi) \right) \\
\left( [p_{m}[0] \neg\varphi] \land \bigwedge_{i \in [1, \text{md}(\varphi)]}(ap)(p_i) \Rightarrow [ap](p_i) \right) \\
\left( \bigwedge_{i \in \{0, \text{md}(\varphi)\}} \left( r \in V[ap] \land [\# = n+1] \land [ap](p_1[0] \neg\varphi) \right) \\
\bigwedge_{i \in \{1, \text{md}(\varphi)\}}(ap)(p_i) \Rightarrow [ap](p_i) \right) \\
\bigwedge_{i \in \{1, \text{md}(\varphi)\}}(ap)(p_i) \Rightarrow [ap](p_i) \right)
\]

Lastly, we define the translation of \( \varphi \), written \( \tau(\varphi) \), into SAL(). It is homomorphic for Boolean connectives and \( \tau(p) \equiv (ap)(p) \) and otherwise it is inductively defined:

\[
\tau(\neg\varphi) = (\text{rel}) \tau(\varphi); \\
\tau(p_1 \varphi_2) = (ap)(p_2) \text{ and otherwise if } \varphi \text{ is satisfiable in ML()}, \text{leading to the following results about the complexity of static ambient logics.}
\]


6.2 Modal separation logic

The family of modal separation logics (MSL), combining separating and modal connectives, has been recently introduced in [21]. Its models, inspired from the memory states used in separation logic (see also [17]), are Kripke-style structures \( \mathfrak{M} = (W, R, V) \), where \( W = \mathbb{N} \) and \( R \subseteq W \times W \) is finite and functional. Hence, unlike finite forests, \( \mathfrak{M} \) may have loops.

Among the fragments studied in [21], the modal separation logic MSL(\( (\ast, \ast)^{-1} \)) was left with a huge complexity gap (between PSpace and Tower). Its formulae are defined from \( \varphi : = p | \bot \varphi \land \varphi | \neg\varphi | \varphi + \varphi \).

The satisfiability relation is as in MSL(\( (\ast, \ast)^{-1} \)) for \( p \in AP \), Boolean connectives and \( \varphi_1 \ast \varphi_2 \), otherwise:

\[
\mathfrak{M}, w \models \varphi \iff \exists w' \text{ s.t. } (w', w) \in R \land \mathfrak{M}, w' \models \varphi.
\]

Since MSL(\( (\ast, \ast)^{-1} \)) is interpreted over a finite and functional relation, \( \ast^{-1} \) effectively works as the \( \diamond \) modality of MSL(). Then, assume we want to check the satisfiability of \( \varphi \) in MSL() by relying on an algorithm for Sat(MSL(\( (\ast, \ast)^{-1} \))). We simply need to consider the formula \( \varphi(\ast \ast \ast \ast^{-1}) \) obtained from \( \varphi \) by replacing every occurrence of \( \ast \) by \( \ast \), and check if it can be satisfied by a locally acyclic model \( (\mathfrak{M}, w) \) of MSL, i.e., one where \( w \) does not belong to a loop of length \( \leq \text{md}(\varphi) \). Local acyclicity can be enforced by the formula

\[
\text{locacycl}_l \equiv \varphi(\ast \ast \ast \ast^{-1}) \land \text{locacycl}_1 \in MSL(\ast, \ast^{-1}) \text{ is satisfiable. Hence, the results in Section 4 allow us to close the complexity gap.}
\]

Corollary 6.2. Sat(MSL(\( (\ast, \ast)^{-1} \))) is Tower-complete.

7 Conclusion

We have studied and compared ML() and MSL(), two modal logics interpreted on finite forests and featuring composition operators. We have not only characterised the expressive power and the complexity for both logics, but also identified remarkable differences and export our results to other logics. ML() is shown as expressive as GML and its satisfiability problem is found to be AExpPol-complete. Besides the obvious similarities between ML() and MSL(), these results are counter-intuitive: though the logic MSL(\( (\ast, \ast) \)) is strictly less expressive than GML and consequently, than ML(), Sat(MSL(\( (\ast, \ast) \))) is Tower-complete. We also recalled that there are logspace reductions from MSL(\( (\ast, \ast) \)) to the second-order modal logic QK\(^t\) from [7].

Our proof techniques go beyond what is known in the literature. For instance, to design the Tower-hardness proof we needed substantial modifications from the proof introduced in [7] for QK\(^t\). On the other hand, to show the expressivity inclusion of MSL() within GML, we provided a novel definition of Ehrenfeucht-Fraïssé games for MSL().

Lastly, our framework led to the characterisation of the satisfiability problems for two sister logics. We proved that the satisfiability problem for the modal separation logic MSL(\( (\ast, \ast)^{-1} \)) is Tower-complete [21]. Moreover, the satisfiability problem for the static ambient logic SAL() is AExpPol-complete, solving open problems from [12, 21] and paving the way to study the complexity of the full SAL.
Acknowledgements
We would like to thank the anonymous reviewers for their comments and suggestions that helped us to improve the quality of the document. B. Bednarczyk is supported by the Polish Ministry of Science and Higher Education program “Diamantowy Grant” no. DI2017 006447. S. Demri and A. Mansutti are supported by the Centre National de la Recherche Scientifique (CNRS). R. Fervari is supported by ANPCyT-PICTs-2017-1130 and 2016-0215, and by the Laboratoire International Associé SINFIN.

References