Submodel Enumeration of Kripke Structures in Modal Logic

Nicolas Fröhlich   Arne Meier

Institut für Theoretische Informatik, Leibniz Universität Hannover
Appelstrasse 9A, 30167 Hannover, Germany
{*nicolas.froehlich, meier*}*@thi.uni-hannover.de*

Abstract

Enumeration complexity (Johnson et al. 1988) is about finding algorithms that produce all solutions to a given problem. Moreover, one strives for a stream of solutions that should be as uniform as possible without much waiting time in between two output solutions and avoiding duplicates. In this paper, we study the problem of model checking in modal logic from this point of view. We consider a particular submodel satisfaction relation that keeps the reference to the transition relation of the original model. Then, we distinguish between enumerating subtrees and subgraphs of Kripke structures that satisfy a modal logic formula. We devise enumeration algorithms for both problems that sort them into the class DelayP.

Keywords: Enumeration Complexity, DelayP, Kripke Structures, Modal Logic.

1 Introduction

In software verification, one strives to know if a written program obeys the underlying specification. It is well-known that describing software systems via a Kripke [21] structure K, i.e., a labelled and state-based transition system, is a profound way in practice [13,30]. For the specification, one constructs an apropriate formula φ in some logic, e.g., temporal [29] or modal logic [3]. Afterwards, in the algorithmic task of model checking [7], one verifies whether K satisfies φ or not. Initially, during the development of a system, the model does not fit the specification yet. On the way to reaching the goal of a system that satisfies the specification formula, considering some kind of restrictions of the program can help in understanding the current issues of the program. Restrictions of the software in turn give rise to submodels of the considered structure K. In this context, not only a particular submodel of K is of much help, as it might be too restrictive; for instance, an empty model could be a strong satisfaction candidate for φ, although not desired. However, a (complete) list of satisfying submodels assists the software developer in adapting the system into the right direction. Obtaining such a list in a systematic way then is a crucial algorithmic task. Moreover, having a uniform stream of printed solutions produced by an algorithm is a key property of a good enumeration process.

Beyond the initial example from above, the described task of model enumeration is, even on the propositional level, very central in many areas, e.g., bounded model checking [2], image computation [15], system engineering [34], and predicate abstraction [22] to name only a few. For a more elaborative view on further applications, we confer the reader to the article of Biere, Möhle and Sebastiani [26].

More formally, in enumeration complexity [19,33], one studies not the overall runtime of an algorithm alone (which often is of exponential duration) but also its *delay*, i.e., an upper bound for all time intervals between two consecutively output solutions (as well as the time before the first and after the last solution has been printed). Here, one strives for algorithms that solve problems within bounds of the class DelayP obeying a delay polynomial in the input length. This class is seen to contain efficient enumeration problems. For modal logic, to the best of the authors' knowledge, a systematic study of enumeration problems in this area has not been undertaken, yet. We want to solve this gap in research, now, and initiate a study of enumeration complexity for model checking in modal logic. We will see that, though having a decision complexity of modal logic model checking in P [35,20], this is not a free ticket for immediately obtaining efficient enumeration algorithms. That is why we start with considering restrictions of the problems in the beginning, namely in the scope of graph restrictions on the obtained submodels.

Contributions. We introduce a family of problems $\mathcal{E}$-ML-SubTree$_N$ that asks for all subtrees of bounded depth $N \in \mathbb{N}$ of a given Kripke structure that satisfy a given formula. We show that for each fixed depth $N \in \mathbb{N}$, the problem can be solved with a delay of $O(|W|^{2N-2} \cdot |\mathrm{SF}(\varphi)|)$, where $|W|$ is the number of states of the given Kripke structure and $|\mathrm{SF}(\varphi)|$ the number of subformulas of the given formula, sorting the problem into the class DelayP. We then show how to improve this result via a recursive approach to reach a delay of $O(|W|^N \cdot |\mathrm{SF}(\varphi)|)$. Consecutively, we tackle the more general version of this problem where all satisfying subgraphs shall be printed. Here, we devise, again, a recursive algorithm that is having a delay of $O(|R|^2 \cdot |W|^2 \cdot |\mathrm{SF}(\varphi)|)$, with $|R|$ the number of transitions in the Kripke structure. We make a rather harsh restriction on what we consider satisfiable subtrees and subgraphs by requiring the $\Box$ operator to be satisfied if and only if all transitions from the original model are present in the subtree or subgraph.

Related work. Krebs et al. [20] investigated the complexity of CTL model checking on the level of operator fragments. This contains a classification of the modal logic variant. There exists a line of research of enumeration complexity in the area of so-called team logics [25,16]. These logics are not built on Kripke but team semantics. Accordingly, their results do not directly transfer to our setting. Capelli and Strozecki [5] study enumeration problems obeying *incremental delay* which could be interesting for extended versions of our studied problems. Also the technique of *geometric amortization* by Capelli and Strozecki [6] might be helpful in this context. Furthermore, a more fine grained enumeration complexity analysis is possible via the framework of parameterised enumeration [24,10,11].

Organisation. First, we will introduce the required foundations of enumeration complexity and briefly present the formalities around modal logic. Then, we will start with the task of subtree enumeration and continue with the generalisation to subgraphs. Finally, we will conclude and present some open research questions.

2 Preliminaries

We assume basic familiarity with computational complexity [28,27].

Modal Logic. We follow the notation of Blackburn et al. [3]. Let PROP be an infinite, countable set of propositions. The set of well-formed formulas $\mathcal{ML}$ is then defined via the following EBNF

$$\varphi \coloneqq p \mid \bot \mid \top \mid \neg\varphi \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \Box\varphi \mid \Diamond\varphi,$$

with $p \in$ PROP. Here, $\top$ is the constant true whereas $\bot$ symbolises the constant false. Let $\varphi \in \mathcal{ML}$ be a formula, then its *length* $|\varphi|$ is defined as the number of its symbols. Let us denote with $\mathrm{SF}(\varphi)$ the set of subformulas of a given formula $\varphi \in \mathcal{ML}$, containing φ as well. Observe that for every $\varphi \in \mathcal{ML}$ we have that $|\mathrm{SF}(\varphi)| \leq |\varphi|$.

Now we turn to Kripke semantics. That is, let $\mathcal{F}$ be a pair (W, R), where W is a non-empty set of *worlds* (or *states*) and $R \subseteq W \times W$ is a binary transition relation on W. The tuple $\mathcal{F}$ is also called a *Kripke structure*. We define a *model* $\mathcal{M}$ to be a pair $(\mathcal{F}, \eta)$, where $\mathcal{F}$ is a Kripke structure, and $\eta\colon \mathrm{PROP} \to \mathcal{P}(W)$ is a map which assigns to each proposition p a set $\eta(p)$ of states. The satisfaction relation is then defined as follows.

Definition 2.1 Let $\varphi, \psi \in \mathcal{ML}$ be two modal formulas. Let $\mathcal{M} = (W, R, \eta)$ be a model and $w \in W$ be a world. We inductively define the *satisfaction* of a formula in the model $\mathcal{M}$ in the world w:

$\mathcal{M}, w \models \top$	always
$\mathcal{M}, w \models \bot$	never
$\mathcal{M}, w \models p$	iff $w \in \eta(p)$ with $p \in$ PROP
$\mathcal{M}, w \models \neg\varphi$	iff $\mathcal{M}, w \not\models \varphi$
$\mathcal{M}, w \models \varphi \wedge \psi$	iff $\mathcal{M}, w \models \varphi$ and $\mathcal{M}, w \models \psi$
$\mathcal{M}, w \models \varphi \vee \psi$	iff $\mathcal{M}, w \models \varphi$ or $\mathcal{M}, w \models \psi$
$\mathcal{M}, w \models \Box\varphi$	iff for all $v \in W$ with $(w, v) \in R$, we have that $\mathcal{M}, v \models \varphi$
$\mathcal{M}, w \models \Diamond\varphi$	iff there exists a $v \in W$ with $(w, v) \in R$ such that $\mathcal{M}, v \models \varphi$

Definition 2.2 Given a modal formula $\varphi \in \mathcal{ML}$, we define its *modal depth* $\mathrm{md}(\varphi)$ in the obvious way:

$$\begin{aligned}
\mathrm{md}(\bot) &\coloneqq \mathrm{md}(\top) \coloneqq \mathrm{md}(p) \coloneqq 0\\
\mathrm{md}(\neg\varphi) &\coloneqq \mathrm{md}(\varphi)\\
\mathrm{md}(\varphi \wedge \psi) &\coloneqq \mathrm{md}(\varphi \vee \psi) \coloneqq \max\{\mathrm{md}(\varphi), \mathrm{md}(\psi)\}\\
\mathrm{md}(\Box\varphi) &\coloneqq \mathrm{md}(\Diamond\varphi) \coloneqq 1 + \mathrm{md}(\varphi)
\end{aligned}$$

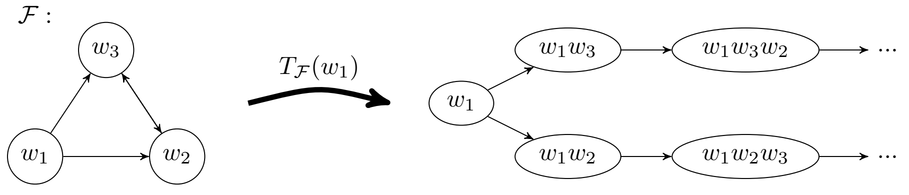

Fig. 1. Example of a computation tree.

Computation Trees. In the following, we define the concept of computation trees that is known from the work of Schnoebelen [31]. Here, a considered Kripke model is unfolded into a computation tree.

First, we need some basic notions. Let $\mathcal{F} = (W, R)$ be a Kripke frame. Furthermore, let a *path (in the Kripke frame $\mathcal{F}$)* $\pi := w_0, w_1, \dots$ be a sequence of worlds with $(w_i, w_{i+1}) \in R$ for $i = 0, 1, \dots$. Here, $\pi_i := w_i$ denotes the i-th element on π, and $\Pi_{\mathcal{F}}(w) := \{\, \pi \mid \pi_0 = w \,\}$ is the set of all paths in $\mathcal{F}$ that start in world w. The following definition is comparable to tree unravelings [4, p.15].

Definition 2.3 Let $\mathcal{F} = (W, R)$ be a Kripke frame and $w \in W$ be a world. The *computation tree (from w in $\mathcal{F}$)* $T_{\mathcal{F}}(w) := (V, E)$ is defined as

$$\begin{aligned} V &:= \left\{\, \pi_0\pi_1 \cdots \pi_{|\pi|} \mid \pi \in \Pi_{\mathcal{F}}(w) \,\right\} \text{ and} \\ E &:= \{\, (\pi, \pi a) \mid \pi, \pi a \in V, a \in W \,\}. \end{aligned}$$

By abuse of notation, if $\mathcal{M} = (\mathcal{F}, \eta)$, we interchangeably use also the notion $T_{\mathcal{M}}(w)$ instead of $T_{\mathcal{F}}(w)$.

Figure 1 depicts an example of a Kripke frame that is unfolded into a computation tree for w_1. Note that often computation trees are objects of infinite size, but also can be finitary. Now, we turn towards the subtree notion.

Definition 2.4 Let $\mathcal{F} = (W, R)$ be a Kripke frame, $w \in W$ a world and $T_{\mathcal{F}}(w) = (V, E)$ be the computation tree from w in $\mathcal{F}$. Then, a *subtree* $T^n_{\mathcal{F}}(w') := (V', E')$ is defined as follows

- $V' \subseteq V$ and $E' \subseteq E$,
- $T^n_{\mathcal{F}}(w')$ is a tree with $w' \in T_{\mathcal{F}}(w)$ as root, and
- $|V'| = n \in \mathbb{N}$ is the number of vertices in the subtree which we also call its *size (of the subtree).*

By abuse of notation, we will also write $T^n_{\mathcal{F}}(w') \subseteq T_{\mathcal{F}}(w)$ for any subtrees of $T_{\mathcal{F}}(w)$. Furthermore, we write $T^n_{\mathcal{F}}(w') \sqsubseteq T_{\mathcal{F}}(w)$, whenever we talk about $T^n_{\mathcal{F}}(w')$ of maximum size.

Notice that in the definition from before, we have $T^m_{\mathcal{F}}(w') \subseteq T^n_{\mathcal{F}}(w)$ for all $m \leq n$, that is, for subtrees with fewer vertices. Also notice that $T^n_{\mathcal{F}}(w') \sqsubseteq T_{\mathcal{F}}(w)$ uniquely determines $T^n_{\mathcal{F}}(w')$.

The *depth* $\operatorname{td}(T^n_{\mathcal{F}})$ of a subtree $T^n_{\mathcal{F}} = (V', E')$ (of a computation tree) then is defined as $\operatorname{td}(T^n_{\mathcal{F}}) := \max_{\pi \in V'}(|\pi|)$, which is the length of its longest path.

The above defined subtrees are a central aspect of this work. They will describe the expected output of the considered enumeration algorithms. We will divide these trees into "satisfiable" and "unsatisfiable" ones. The following satisfaction definition of subtrees will contain a reference to the transition relation of the original structure for the modal operators. We give more insights (see Example 3.7 for this) on this after the presentation of our enumeration algorithm that is used to prove Theorem 3.5.

Definition 2.5 Let $\mathcal{M} = (W, R, \eta)$ be a model, $\varphi \in \mathcal{ML}$ be a formula and $T^n_{\mathcal{M}}(w) = (V', E')$ be a subtree of the computation tree in $w \in W$. Then let $T^n_{\mathcal{M}}(w) \models \varphi$ be inductively defined as

$$\begin{array}{lll}
T^n_{\mathcal{M}}(w) \models \varphi & \text{iff} & \mathcal{M}, w \models \varphi \text{ for } \varphi \in \{\top, \bot\} \cup \mathrm{PROP}, \\
T^n_{\mathcal{M}}(w) \models \neg\varphi & \text{iff} & T^n_{\mathcal{M}}(w) \not\models \varphi, \\
T^n_{\mathcal{M}}(w) \models \varphi \wedge \psi & \text{iff} & T^n_{\mathcal{M}}(w) \models \varphi \text{ and } T^n_{\mathcal{M}}(w) \models \psi, \\
T^n_{\mathcal{M}}(w) \models \varphi \vee \psi & \text{iff} & T^n_{\mathcal{M}}(w) \models \varphi \text{ or } T^n_{\mathcal{M}}(w) \models \psi, \\
T^n_{\mathcal{M}}(w) \models \Diamond\varphi & \text{iff} & \exists w_0 \text{ with } (w, w_0) \in R : \text{for } T^m_{\mathcal{M}}(w_0) \sqsubseteq T^n_{\mathcal{M}}(w) \\
 & & \text{we have that } (w, ww_0) \in E' \text{ and } T^m_{\mathcal{M}}(w_0) \models \varphi, \\
T^n_{\mathcal{M}}(w) \models \Box\varphi & \text{iff} & \forall w_0 \text{ with } (w, w_0) \in R : \text{for } T^m_{\mathcal{M}}(w_0) \sqsubseteq T^n_{\mathcal{M}}(w) \\
 & & \text{we have that } (w, ww_0) \in E' \text{ and } T^m_{\mathcal{M}}(w_0) \models \varphi.
\end{array}$$

Notice that for the modal operators $\Box/\Diamond$, we require that for every/one original edge in (W, R) there exists a satisfying subtree. Because of this, it is not possible that $\mathcal{M}, w \not\models \Box\psi$ but a subtree satisfies $\Box\psi$. Clearly, trying to "hide" $\Box$ operators with $\neg\Diamond\neg$ does not work. The reason for that is that for both operators the satisfaction relation for subtrees is defined with respect to the original model. Currently, we need this restriction to limit the number of possibilities in the constructed enumeration algorithms below. We will leave it as a question for future research, whether there exists still a polynomial delay algorithm solving the enumeration problem with an unrestricted satisfaction relation for subtrees.

Enumeration Complexity. In contrast to decision problems, which ask for the existence of solutions to a given instance, for enumeration problems one is concerned with the output of *all* solutions to an instance. Since the number of solutions is usually of exponential size, the running time between the output of two solutions is of particular interest. This elapsed time interval is called the *delay* of an enumeration algorithm and will be defined shortly.

In this context, random access machines are often chosen as the computational model. In this paper, we will use the common one [33,9]. Note that, in this model, one can access particular parts of exponentially large priority queues in polynomial time [19]. The definitions of enumeration problems will follow also recent standard terminology [33,9,24].

Definition 2.6 An *enumeration problem (EP)* is a tuple $\mathcal{E} = (I, \mathrm{Sol})$, where

- I is the set of *instances*, and

- Sol is a function such that for all $x \in I$ the set $\mathrm{Sol}(x)$ is the finite set of *solutions (of x)*.

In this paper, the studied enumeration problems all have the property that we always have a fixed polynomial p such that for every instance $x \in I$ and every solution $y \in \mathrm{Sol}(x)$, we have that $|y| \leq p(|x|)$. Such problems are often classified by a class that is called EnumP [33]. In some sense, this class can be seen as a natural counterpart to NP in the classical setting. Now, we are ready to define enumeration algorithms.

Definition 2.7 Let $\mathcal{E} = (I, \mathrm{Sol})$ be an EP. An algorithm $\mathcal{A}$ is said to be an *enumeration algorithm (EA)* for $\mathcal{E}$, if for every $x \in I$ the algorithm $\mathcal{A}$ obeys the following two properties, where $\mathcal{A}(x)$ denotes the *computation of $\mathcal{A}$ on input x*:

- $\mathcal{A}(x)$ terminates after a finite sequence of steps.
- $\mathcal{A}(x)$ prints exactly $\mathrm{Sol}(x)$ without duplicates.

In the next result, we formally define the notion of a delay of an enumeration algorithm.

Definition 2.8 Let $\mathcal{E} = (I, \mathrm{Sol})$ be an EP, $\mathcal{A}$ be an EA for $\mathcal{E}$, and $x \in I$ be an instance. Then we define

- the *ith delay* of $\mathcal{A}(x)$ as the elapsed time between the output of the ith and $(i+1)$st solution of $\mathrm{Sol}(x)$ by $\mathcal{A}$ on input x,
- the *0th delay* as the *precomputation time*, i.e, the elapsed time before the first output of $\mathcal{A}(x)$, and
- the *mth delay* as the *postcomputation time*, i.e., the elapsed time after the last output of $\mathcal{A}(x)$ until it terminates.

We say that $\mathcal{A}$ has *delay $t(n)$*, for some function $t\colon \mathbb{N} \to \mathbb{N}$, if for all $x \in I$ and all $0 \leq i \leq m$ the ith delay of $\mathcal{A}(x)$ is in $O(t(|x|))$.

After having defined the formalisms around the machine model of enumeration complexity, we will define the complexity class that is relevant in this paper.

Definition 2.9 Let $\mathcal{E} = (I, \mathrm{Sol})$ be an EP and $\mathcal{A}$ be an EA for $\mathcal{E}$. If there exists a polynomial p such that $\mathcal{A}$ has delay $p(n)$, then E belongs to the complexity class DelayP.

3 Enumeration of Subtrees

Before we consider enumeration in modal logic, we first want to take a look at a result by Vardi [35], resp., Clarke and Allen Emerson [8], which shows that model checking for modal formulas can be done in polynomial time.

Proposition 3.1 ([35, Prop. 2.1],[8]) *Let $\mathcal{M} = (W, R, \eta)$ be a model, $w \in W$ a world and φ be a modal formula. Then model checking, i.e., checking whether $(M, w) \models \varphi$ is true, can be verified in time $O(|W|^2 \cdot |\mathrm{SF}(\varphi)|)$.*

Proof. Let $\varphi_1, \dots, \varphi_m \in \mathrm{SF}(\varphi)$ be the subformulas of φ, for some $m \in \mathbb{N}$, listed in increasing order of length, with ties broken arbitrarily. As a result, we have that $\varphi_m = \varphi$ and if φ_i is a proper subformula of φ_j, then $i < j$. There are at most $|\varphi|$-many subformulas of φ, so we must have that $m \leq |\varphi|$. An induction over k shows that we can label each world $w \in W$ with φ_j or $\neg\varphi_j$, for $j = 1, \dots, k$, depending on whether or not φ_j is true at w, in time $O(k \cdot |W|)$. The only nontrivial cases are if φ_{k+1} is of the form $\Box\varphi_j$ or $\Diamond\varphi_j$, where $j < k + 1$. We label a world w with $\Box\varphi_j$ if and only if each world t such that $(w, t) \in R$ is labelled with φ_j, and with $\Diamond\varphi_j$ if and only if there is a world t such that $(w, t) \in R$ is labelled with φ_j. Assuming inductively that each state has already been labelled with φ_j or $\neg\varphi_j$, this step can clearly be carried out in time $O(|W|^2)$. The total time required is accordingly bound by $O(|W|^2 \cdot |\mathrm{SF}(\varphi)|)$ as desired. □

It is easy to see that this can be used to determine the satisfiability of subtrees $T^n_{\mathcal{M}}(w)$, as they can be seen as acyclic Kripke models themselves. We will now show that the labelling introduced above can easily be updated when removing a leaf from a subtree.

Lemma 3.2 *Let $\mathcal{M} = (W, R, \eta)$ be a model, $w \in W$ be a world, and φ be a modal formula. Given a labelling of satisfied subformulas for all nodes of a subtree $T^n_{\mathcal{M}}(w)$ and subformulas* $\mathrm{SF}(\varphi)$, *we can correctly update the labelling after removing one leaf in time $O(\mathrm{td}(T^n_{\mathcal{M}}(w)) \cdot |W| \cdot |\mathrm{SF}(\varphi)|)$.*

Proof. Let φ_i or $\neg\varphi_i$ be the labelling for each node in $T^n_{\mathcal{M}}(w)$ and subformulas $\varphi_i \in \mathrm{SF}(\varphi)$. Also let $\ell = \pi_0\pi_1 \dots \pi_j$ be the leaf removed from $T^n_{\mathcal{M}}(w)$. It should be clear that only nodes $\pi_0\pi_1 \dots \pi_{j-k}$, with $1 \leq k \leq j$, on the path from root w to leaf ℓ can be effected by the removal. Each of these nodes will have to update $|\mathrm{SF}(\varphi)|$ labels and must check at most $|W|$ nodes for labels $\Box\varphi_i$ and $\Diamond\varphi_i$. The maximum length of a path in $T^n_{\mathcal{M}}(w)$ cannot exceed $\mathrm{td}(T^n_{\mathcal{M}}(w))$. Together the update requires a time of $O(\mathrm{td}(T^n_{\mathcal{M}}(w)) \cdot |W| \cdot |\mathrm{SF}(\varphi)|)$. □

Let us start with a problem that immediately lifts model checking to the enumeration setting.

Problem:	$\mathcal{E}$-ML-SubTree
Input:	$(\mathcal{M}, w, \varphi)$, model $\mathcal{M} = (W, R, \eta)$, world $w \in W$, formula φ
Output:	All subtrees of $T_{\mathcal{M}}(w)$ that satisfy φ

Unfortunately, the next theorem shows that this problem is not an EP as defined in Def. 2.6 as it violates the second property.

Theorem 3.3 *There exists an input $x := (\mathcal{M}, w, \varphi)$ to* $\mathcal{E}$-ML-SubTree *such that $|\mathrm{Sol}(x)| = \infty$.*

Proof. Consider the rather simple input

$$x = ((\{w\}, \{(w, w)\}, \eta), w, \top).$$

The formula $\varphi = \top$ is trivially satisfied in all possible subtrees of $T_{\mathcal{M}}(w)$. As a result, we can construct an infinite number of subtrees $T^n_{\mathcal{M}}(w) \subseteq T_{\mathcal{M}}(w)$, from which we deduce that $|\mathrm{Sol}(x)|$ is infinite. □

Now, we consider a restriction of the previous problem motivated by the last result. Notice that the depth bound $N \in \mathbb{N}$ is part of the problem definition and not part of the instance.

Problem:	$\mathcal{E}$-ML-SubTree$_N$, where $N \in \mathbb{N}$.
Input:	$(\mathcal{M}, w, \varphi)$, model $\mathcal{M} = (W, R, \eta)$, world $w \in W$, formula φ
Output:	All subtrees of $T_{\mathcal{M}}(w)$ with $\mathrm{td}\,(T^n_{\mathcal{M}}(w)) \leq N$ that satisfy φ

We want to mention that on the level of debugging, as explained in the introduction, one usually has that φ is not satisfied by $\mathcal{M}$ in w. This would be a different enumeration problem which asks for falsifying subtrees (to satisfy the formula again which was initially not satisfied). Asking for falsified submodels intuitively makes the problem much harder as we have no real limit on the search space. That is why we currently do not consider this problem and leave this question for future research.

In the following, we show how utilising a priority queue will lead to an enumeration algorithm obeying a polynomial delay. Note that the technique of priority queues goes back to Johnson, Papadimitriou and Yannakakis [19]. As this approach has recurred, the question arose whether the downside of a possibly exponential space can be circumvented [5,6]. The technique of geometric amortization suggested by Capelli and Strozecki [6] can find here an application as well.

Theorem 3.4 *For all fixed $N \in \mathbb{N}$, we have that $\mathcal{E}$-ML-SubTree$_N$ $\in$ DelayP. More precisely, there exists an enumeration algorithm for $\mathcal{E}$-ML-SubTree$_N$ having a delay of $O(|W|^{2N-2} \cdot |\mathrm{SF}(\varphi)|)$, where (W, R, η) is the input model and φ is the given formula.*

Proof. Algorithm 1 uses a priority queue to systematically process the largest subtrees first and avoid printing duplicates. Note, that the largest subtree is uniquely determined.

Let us first turn towards the correctness of the algorithm. Clearly, the algorithm only outputs subtrees which satisfy φ, since only such subtrees are added into the priority queue. In case that the largest subtree $T^n_{\mathcal{M}}(w)$ with depth N does not satisfy φ, the algorithm already does not enter the **if**-condition in line 3 and outputs nothing. Since the size of subtrees added to Q decreases monotonically the algorithm will eventually terminate when there are no more leaves to cut. In line 9, we also only insert subtrees into Q which we have not seen already preventing outputting duplicates. Lastly, the algorithm will reach all eligible subtrees, because it only stops to consider subtrees, that do not satisfy φ. Notice that subtrees of already unsatisfiable subtrees cannot become satisfiable by removing additional nodes (confer Def. 2.5). To summarise, the algorithm will terminate, output all satisfiable subtrees, and avoid printing duplicates.

Now, we classify the delay of the algorithm. Initialising the priority queues S and Q requires constant time. The number of nodes in the largest subtree $T^n_{\mathcal{M}}(w)$ with depth N is bound by $\sum_{i=0}^{N-1}|W|^i$. This bound is reached only for fully connected Kripke structures. In that case, each world has $|W|$ outgoing relations. As a result, the computation tree consists of a root with $|W|$ children, each having $|W|$ children and so forth. The total sum of nodes in such a subtree, with depth N is, accordingly,

$$|W|^0 + |W|^1 + \cdots + |W|^{N-1} = \sum_{i=0}^{N-1} |W|^i = \frac{|W|^N - 1}{|W| - 1} \in O(|W|^{N-1}).$$

Since worlds in not fully connected Kripke structures have fewer outgoing transitions, their computation trees and subtrees will consist of even less nodes. In the worst case, the time required to create the largest subtree $T^n_{\mathcal{M}}(w)$ with depth N in line 2 is thereby bounded by $O(|W|^{N-1})$. The model checking task requires polynomial time (see Prop. 3.1) and line 3 can be done in time $O(|W|^{N-1} \cdot |W| \cdot |\mathrm{SF}(\varphi)|)$, with $|\mathrm{SF}(\varphi)|$ the number of labels per node, $|W|^{N-1}$ the total number of nodes, and $|W|$ the maximum number of worlds considered for subformulas $\Box\varphi_i$ or $\Diamond\varphi_i$. The delay between two consecutively output solutions is now determined by the time of each **while**-iteration. Extracting, outputting, and adding a subtree to S can all be done in time $O(|W|^{N-1})$. The number of leaves with a maximum depth N is equal to or less than $|W|^{N-1}$, therefore line 7 loops for $\leq |W|^{N-1}$ times. In each iteration, we have to model check the new subtree and Lemma 3.2 shows that the time needed in $O(N \cdot |W| \cdot |\mathrm{SF}(\varphi)|)$. Note that N is not part of the input and can therefore be omitted in the following. Checking $T^{n-1}_{\mathcal{M}} \notin Q \cup S$ and potentially adding to Q will still require a time of $O(|W|^{N-1})$. Together, we have an upper bound of

$$O(|W| \cdot |\mathrm{SF}(\varphi)| + |W|^{N-1}) \subseteq O(|W|^{N-1} \cdot |\mathrm{SF}(\varphi)|)$$

for one iteration of the **for**-loop.

Overall the delay is in

$$O\big(|W|^{N-1} \cdot |W|^{N-1} \cdot |\mathrm{SF}(\varphi)|\big) = O\big(|W|^{2N-2} \cdot |\mathrm{SF}(\varphi)|\big)\,,$$

which shows the desired DelayP result. □

In the next step, we will explain how to improve the algorithm from before yielding an EA with a faster delay.

Theorem 3.5 *For all $N \in \mathbb{N}$, there exists an enumeration algorithm for $\mathcal{E}$-ML-SubTree$_N$ having a delay of $O(|W|^N \cdot |\mathrm{SF}(\varphi)|)$, where (W, R, η) is the input model and φ is the given formula.*

Proof. In the following, we will explain how Alg. 2 achieves the desired improvement. Let $T^n_{\mathcal{M}}(w)$ be the largest subtree given a model $\mathcal{M}$ and world w. Furthermore, let $T^m_{\mathcal{M}}(w)$ be any subtree of $T^n_{\mathcal{M}}(w)$ that satisfies $T^m_{\mathcal{M}}(w) \models \varphi$ for a given modal formula φ. Now, we define the set M as follows:

$$M := \{\, k \mid k \text{ is a node in } T^n_{\mathcal{M}}(w), \text{ but not in } T^m_{\mathcal{M}}(w) \,\}.$$

Algorithm 1: Enumeration algorithm for $\mathcal{E}$-ML-SubTree$_N$.

```
Input: Model $\mathcal{M} = (W, R, \eta)$, world $w \in W$ and formula $\varphi$
1  initialise priority queues $S \leftarrow \emptyset$ and $Q \leftarrow \emptyset$
2  add largest subtree $T^n_{\mathcal{M}}(w) \sqsubseteq T_{\mathcal{M}}(w)$ with $\operatorname{td}(T^n_{\mathcal{M}}(w)) \leq N$ to $Q$
3  if $T^n_{\mathcal{M}}(w) \models \varphi$ then
4      while $Q \neq \emptyset$ do
5          extract largest subtree $T^n_{\mathcal{M}}(w)$ from $Q$
6          output $T^n_{\mathcal{M}}(w)$ and add it to $S$
7          for all $T^{n-1}_{\mathcal{M}} \subseteq T^n_{\mathcal{M}}(w)$ do
8              if $T^{n-1}_{\mathcal{M}}(w) \models \varphi$ and $T^{n-1}_{\mathcal{M}}(w) \notin Q \cup S$ then
9                  insert $T^{n-1}_{\mathcal{M}}(w)$ into $Q$
10 terminate
```

Algorithm 2: Direct enumeration for $\mathcal{E}$-ML-SubTree$_N$.

```
Input: Model $\mathcal{M} = (W, R, \eta)$, world $w \in W$ and formula $\varphi$
1  determine the largest subtree $T^n_{\mathcal{M}}(w) \sqsubseteq T_{\mathcal{M}}(w)$ with $\operatorname{td}(T^n_{\mathcal{M}}(w)) \leq N$
2  label nodes in $T^n_{\mathcal{M}}(w)$ in BFS order
3  if $T^n_{\mathcal{M}}(w) \models \varphi$ then
4      Output $T^n_{\mathcal{M}}(w)$
5      for all leaves $\ell$ in $T^n_{\mathcal{M}}(w)$ do
6          EnumSubtreeRec($T^n_{\mathcal{M}}(w), \varphi, \ell$)
7  Function EnumSubtreeRec($T(w), \varphi, k$):
8      remove leaf $k$ from $T(w)$
9      if $T(w) \models \varphi$ then
10         Output $T(w)$
11         for all leaves $\ell$ in $T(w)$ with $\ell <_{\mathrm{BFS}} k$ in descending order do
12             EnumSubtreeRec($T(w), \varphi, \ell$)
```

The elements in M are the leaves removed by the recursive calls of EnumSubtreeRec. Because leaves are only removed in descending order there is a unique path from $T^n_{\mathcal{M}}(w)$ to $T^m_{\mathcal{M}}(w)$. Also, if we have that $T^m_{\mathcal{M}}(w) \models \varphi$, then the same is true for all subtrees between $T^n_{\mathcal{M}}(w)$ and $T^m_{\mathcal{M}}(w)$, as adding nodes cannot make a subtree falsifying.

Now, we measure the delay of Alg. 2. First, the algorithm constructs the largest subtree $T^n_{\mathcal{M}}(w)$ with depth N, in line 1, which we have seen in the proof of Theorem 3.4 to be bounded by $O(|W|^{N-1})$. Likewise, the labelling of nodes (line 2) and outputting (line 4) can be done in time $O(|W|^{N-1})$, while checking satisfiability (line 3) requires a time of $O(|W|^N \cdot |\mathrm{SF}(\varphi)|)$. Because we immediately output a subtree, if it is satisfiable, the worst delay occurs when iterating over unsatisfiable subtrees. The maximum number of unsatisfiable

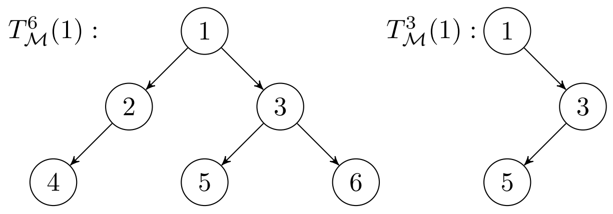

Fig. 2. $M = \{6, 4, 2\}$

subtrees will be in the initial FOR-loop in line 5, with no more than $|W|^{N-1}$ iterations. For each examined subtree we need to check, if it is still satisfiable after removing a leaf in line 8. We have shown in Lemma 3.2 that this can be achieved for each subtree in time $O(N \cdot |W| \cdot |\mathrm{SF}(\varphi)|)$, and again we can omit the N since it is not part of the input. It is also not necessary to consider the depth of recursion here, because if we would find satisfiable subtrees $T^m_{\mathcal{M}}(w)$, all subtrees $T^n_{\mathcal{M}}(w)$ with $T^m_{\mathcal{M}}(w) \subseteq T^n_{\mathcal{M}}(w)$ would be guarantied to be satisfiable and thus reduce the delay. Together the delay of Algorithm 2 is

$$O(|W|^{N-1} \cdot |W| \cdot |\mathrm{SF}(\varphi)|) = O(|W|^N \cdot |\mathrm{SF}(\varphi)|),$$

which shows the desired result. $\square$

The next example illustrates Alg. 2 and the proof of Theorem 3.5.

Example 3.6 In Figure 2, we can see a simplified versions of the subtree $T^6_{\mathcal{M}}(1)$ and of its subtree $T^3_{\mathcal{M}}(1)$. The nodes $6, 4, 2$ are missing from $T^3_{\mathcal{M}}(w)$ and are collected in the set M. The elements from M in descending order show the corresponding calls to EnumSubtreeRec(6), EnumSubtreeRec(4) and EnumSubtreeRec(2) to go from $T^6_{\mathcal{M}}(1)$ to $T^3_{\mathcal{M}}(w)$.

The following example gives more justification for our decision on incorporating the reference to the original transition relation in Definition 2.5. Intuitively, allowing to cut subtrees opens the (problematic) possibility that too many subtrees have to be considered (for instance, in case a clique appears here) that all are not satisfying until the dead-end for the $\Box$ is reached.

Example 3.7 Let $\mathcal{M} = (W, R, \eta)$ be a model with

$$\begin{aligned} W &:= \{w, w_0\} \cup \{w_i \mid 1 \le i \le k\}, \\ R &:= \{(w, w_0), (w_0, w_1)\} \cup \{(w_i, w_j) \mid 1 \le i, j \le k\}, \\ \eta &:= \{x \mapsto \{w\}\}. \end{aligned}$$

Let $\varphi := \Box x$, and replace in Def. 2.5 the semantics of subtree satisfaction for the modal operators with the following

$$\begin{aligned} T^n_{\mathcal{M}}(w) \models \Diamond\varphi \quad &\text{iff} \quad \exists w_0 \text{ with } (w, ww_0) \in E' : \text{for } T^m_{\mathcal{M}}(w_0) \sqsubseteq T^n_{\mathcal{M}}(w) \\ &\qquad \text{we have that } T^m_{\mathcal{M}}(w_0) \models \varphi, \\ T^n_{\mathcal{M}}(w) \models \Box\varphi \quad &\text{iff} \quad \forall w_0 \text{ with } (w, ww_0) \in E' : \text{for } T^m_{\mathcal{M}}(w_0) \sqsubseteq T^n_{\mathcal{M}}(w) \\ &\qquad \text{we have that } T^m_{\mathcal{M}}(w_0) \models \varphi, \end{aligned}$$

where $T^n_{\mathcal{M}}(w) = (V', E')$.

Then, Algorithm 2 is no enumeration algorithm, as it does not consider the subtree where $T^m_{\mathcal{M}}(w_0)$ is cut off. The reason for that is found in line 3 and 9 as without cutting off $T^m_{\mathcal{M}}(w_0)$, the subtree is not satisfying, yet.

Further notice that analysing the delay of a modification of Algorithm 2 without the restrictions in line 3 and 9 and instead applying the requirement of satisfiability only onto line 4 and 10 directly to prevent printing unsatisfiable subtrees, also does not perform as desired. To see this, first notice that $T^n_{\mathcal{M}}(w) \not\models \varphi$ unless $T^m_{\mathcal{M}}(w_0) \sqsubseteq T^n_{\mathcal{M}}(w)$ is cut from it, because $w_0 \notin \eta(x)$. Also consider that our algorithm starts with the largest possible subtree and successively removes leaves. Since w_0 is the root of $T^m_{\mathcal{M}}(w_0)$ it will naturally be removed last. We therefore have to look at the number of unsatisfiable subtrees the algorithm considers before w_0 becomes a leaf. Observe that $\{\, w_i \mid 1 \leq i \leq k \,\}$ and $\{\, (w_i, w_j) \mid 1 \leq i, j \leq k \,\}$ result in a fully connected subgraphs with k worlds. This will create a full k-ary subtree, which the algorithm as to iterate through. The number of full k-ary trees is recursively given by $a(n+1) = a(n)^k + 1$, with $a(0) = 0$ (this bound follows by a straightforward inductive proof on n and k). It follows that for $a(3) = 2^k + 1$ the number of subtrees is already exponentially in the input size, namely the number of worlds $|W|$ of $\mathcal{M}$ which results in a delay between the largest subtree and the first satisfiable subtree, yielding an exponential delay.

4 Enumeration of subgraphs

Finally, we consider generalising the problem $\mathcal{E}$-ML-SubTree$_N$ to arbitrary subgraphs that do not necessarily have to be trees and also remove the depth bound. By this, we also leave the notion of computation trees that talk about unfolding of a Kripke structure as for the subtree notion. Given a Kripke frame $\mathcal{F} = (W, R)$, a *subgraph* $S_{\mathcal{F}}(R')$ of (W, R) is simply considered with respect to subsets of the edge set $R' \subseteq R$. Furthermore, similar as in Def. 2.5, we define satisfaction of subgraphs. Note that, here again, we need the property regarding modality-prefixed formulas that refer back to the original transition relation R.

Problem:	$\mathcal{E}$-ML-SubGraph
Input:	$(\mathcal{M}, w, \varphi)$, $\mathcal{M} = (W, R, \eta)$ model, φ formula
Output:	All subgraphs of $\mathcal{M}$ that satisfy φ

Theorem 4.1 $\mathcal{E}$-ML-SubGraph $\in$ DelayP.

Proof. Algorithm 3 enumerates all solutions of a model by recursively removing transitions until a resulting subgraphs no longer satisfies the given formula φ. Labelling the transitions uniquely allows us to guarantee no duplicate outputs. As a result, the algorithm is assured to visit each subgraph of $\mathcal{M}$ at most once, analogically to the proof of Theorem 3.5. Subgraphs are not visited, only if a subgraph before already does not satisfy φ, which means that unvisited subgraphs cannot satisfy φ, too. This results in an output of all subgraphs of

Algorithm 3: Subgraph enumeration.

```
   Input: Model $\mathcal{M} = (W, R, \eta)$ and formula $\varphi$
1  label transitions $r \in R$ uniquely
2  if $S_{\mathcal{M}}(R) \models \varphi$ then
3      Output $S_{\mathcal{M}}(R)$
4      for all transitions $r \in R$ do
5          EnumSubgraphRec($\mathcal{M}, \varphi, \{r\}$)
6  Function EnumSubgraphRec($\mathcal{M}, \varphi, R'$):
7      if $S_{\mathcal{M}}(R \setminus R') \models \varphi$ then
8          Output $\mathcal{M}$
9          for all transitions $r_0 \in R \setminus R'$ with $r_0 < r$ do
10             EnumSubgraphRec($\mathcal{M}, \varphi, R' \cup \{r_0\}$)
```

$\mathcal{M}$ that satisfy φ without duplicates.

We have already established, that model checking can be done in polynomial time (see Prop. 3.1). Outputting and modifying subgraphs can all be done in linear time, with respect to the size of the initial model. (For the following worst case estimation, also see Example 4.2 below.) The worst case happens if firstly a sequence of EnumSubgraphRec calls all result in subgraphs, that satisfy φ, which can be at most $|R|$, where $\mathcal{M} = (W, R, \eta)$ is the input instance. Secondly, all remaining calls of EnumSubgraphRec in all prior recursion steps result in subgraphs not satisfying φ, which is bounded by the maximum recursion depth $|R|$ and the number of possible subgraphs at each step which is $|R| - d$ for d the current depth of the recursion. Together, we have that the number of unsatisfying subgraphs to be checked before we can terminate has to be less than

$$\sum_{i=0}^{|R|} |R| - i = \frac{|R|^2 + |R|}{2} \in O(|R|^2)$$

Each subgraph has to be model checked, which is done in time $O(|W|^2 \cdot |\mathrm{SF}(\varphi)|)$ each, resulting in a total delay of

$$O(|R|^2 \cdot |W|^2 \cdot |\mathrm{SF}(\varphi)|).$$

With this we have shown that the delay of Alg. 3 is requiring polynomial time and, accordingly, $\mathcal{E}$-ML-SubGraph $\in$ DelayP is true. □

Example 4.2 Let $\mathcal{M}$ be a Kripke structure with $R = \{r_0, r_1, r_2, r_3\}$ the set of transitions labelled by their index. Each edge in Fig. 3 represents a call to EnumSubgraphRec, and nodes are labelled with the transitions of the current subset. The algorithm firstly finds a sequence of satisfiable models going deeper in its recursion until it reaches only unsatisfiable models. It then has to model check all remaining candidates which totals in

$$7 < 10 = \frac{|R|^2 + |R|}{2}.$$

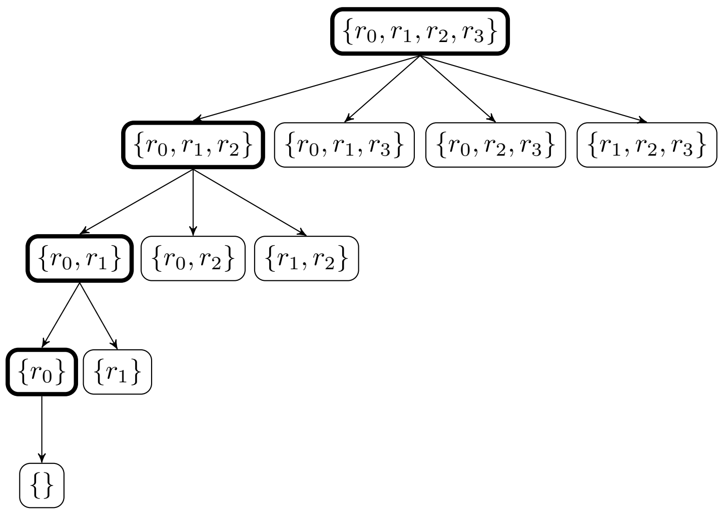

Fig. 3. Recursive calls to EnumSubgraphRec and their resulting transition sets. Nodes marked by a thicker border are satisfiable, while thin borders denote unsatisfiable ones.

This is obviously less than our given upper bound in the proof of Theorem 4.1.

5 Conclusion

In this paper, we classified the enumeration complexity of two problems located in modal logic. First, we studied the family of problems $\mathcal{E}$-ML-SubTree$_N$ that asks, for each $N \in \mathbb{N}$, to print all satisfying subtree models restricted to a depth N of a given Kripke model and showed that this can be efficiently done with a polynomial delay, placing the enumeration problem in the class DelayP. In particular, we showed two algorithmic ways to obtain this result and eventually presented a more efficient version which has a delay of $O(|W|^N \cdot |\mathrm{SF}(\varphi)|)$, where W is the set of worlds of the given Kripke model and $\mathrm{SF}(\varphi)$ are the subformulas of the given modal formula. In this context, it might be worth studying the framework of parameterised complexity [11,10,24] to find more efficient algorithms under possibly the tree-width [12, Cha. 7] parameterisation. Particularly, space-efficient algorithms could be of interest here and placing such problems into classes with very efficient space notions would be very tempting [17]. Another possibility would be to utilise the framework of hard enumeration [9] to show negative answers to this question. Also, a completely different angle on the problems studied here is on the level of parameterised counting [23,14].

Second, we considered a generalisation of this problem, removing the fixed depth bound and asking for satisfying subgraphs (that do not necessarily need to be trees). We showed that an enumeration algorithm exists with a delay of $O(|R|^2 \cdot |W|^2 \cdot |\mathrm{SF}(\varphi)|)$.

An obvious next step would be to investigate ways to weaken our restriction on the satisfiability of the modal operators in subtrees and subgraphs and then still obtain an enumeration algorithm without exponential delay. Also taking a look at other problems, e.g., submodels on the level of subsets of the labelling function η as well as practical implementations and integrations into recent model checkers [18,32,1] might yield interesting insights and results.

Acknowledgements

The authors thank the anonymous reviewers for their valuable comments.

References

[1] Bengtsson, J., K. G. Larsen, F. Larsson, P. Pettersson and W. Yi, *UPPAAL - a tool suite for automatic verification of real-time systems*, in: *Hybrid Systems*, Lecture Notes in Computer Science **1066** (1995), pp. 232–243.

[2] Biere, A., A. Cimatti, E. M. Clarke, O. Strichman and Y. Zhu, *Bounded model checking*, Adv. Comput. **58** (2003), pp. 117–148.

[3] Blackburn, P., M. de Rijke and Y. Venema, "Modal Logic," Cambridge Tracts in Theoretical Computer Science **53**, Cambridge University Press, 2001.

[4] Blackburn, P. and J. van Benthem, *Modal logic: a semantic perspective*, in: P. Blackburn, J. F. A. K. van Benthem and F. Wolter, editors, *Handbook of Modal Logic*, Studies in logic and practical reasoning **3**, North-Holland, 2007 pp. 1–84.
URL `https://doi.org/10.1016/s1570-2464(07)80004-8`

[5] Capelli, F. and Y. Strozecki, *Incremental delay enumeration: Space and time*, Discret. Appl. Math. **268** (2019), pp. 179–190.

[6] Capelli, F. and Y. Strozecki, *Geometric amortization of enumeration algorithms*, CoRR **abs/2108.10208** (2021).

[7] Clarke, E. M., *The birth of model checking*, in: *25 Years of Model Checking*, Lecture Notes in Computer Science **5000** (2008), pp. 1–26.

[8] Clarke, E. M. and E. A. Emerson, *Design and synthesis of synchronization skeletons using branching-time temporal logic*, in: *Logic of Programs*, Lecture Notes in Computer Science **131** (1981), pp. 52–71.

[9] Creignou, N., M. Kröll, R. Pichler, S. Skritek and H. Vollmer, *A complexity theory for hard enumeration problems*, Discret. Appl. Math. **268** (2019), pp. 191–209.

[10] Creignou, N., R. Ktari, A. Meier, J. Müller, F. Olive and H. Vollmer, *Parameterised enumeration for modification problems*, Algorithms **12** (2019), p. 189.

[11] Creignou, N., A. Meier, J. Müller, J. Schmidt and H. Vollmer, *Paradigms for parameterized enumeration*, Theory Comput. Syst. **60** (2017), pp. 737–758.

[12] Cygan, M., F. V. Fomin, L. Kowalik, D. Lokshtanov, D. Marx, M. Pilipczuk, M. Pilipczuk and S. Saurabh, "Parameterized Algorithms," Springer, 2015.

[13] Fix, L., *Fifteen years of formal property verification in Intel*, in: *25 Years of Model Checking*, Lecture Notes in Computer Science **5000** (2008), pp. 139–144.

[14] Flum, J. and M. Grohe, *The parameterized complexity of counting problems*, in: *FOCS* (2002), p. 538.

[15] Gupta, A., Z. Yang, P. Ashar and A. Gupta, *Sat-based image computation with application in reachability analysis*, in: *FMCAD*, Lecture Notes in Computer Science **1954** (2000), pp. 354–371.

[16] Haak, A., A. Meier, F. Müller and H. Vollmer, *Enumerating teams in first-order team logics*, CoRR **abs/2006.06953** (2020).

[17] Haak, A., A. Meier, O. Prakash and B. V. R. Rao, *Parameterised counting in logspace*, in: *STACS*, LIPIcs **187** (2021), pp. 40:1–40:17.

[18] Holzmann, G. J., "The SPIN Model Checker - primer and reference manual," Addison-Wesley, 2004.

[19] Johnson, D. S., C. H. Papadimitriou and M. Yannakakis, *On generating all maximal independent sets*, Inf. Process. Lett. **27** (1988), pp. 119–123.

[20] Krebs, A., A. Meier and M. Mundhenk, *The model checking fingerprints of CTL operators*, Acta Informatica **56** (2019), pp. 487–519.

[21] Kripke, S., *Semantical considerations on modal logic*, Acta Philosophica Fennica **16** (1963), pp. 83–94.

[22] Lahiri, S. K., R. Nieuwenhuis and A. Oliveras, *SMT techniques for fast predicate abstraction*, in: *CAV*, Lecture Notes in Computer Science **4144** (2006), pp. 424–437.

[23] McCartin, C., *Parameterized counting problems*, in: *MFCS*, Lecture Notes in Computer Science **2420** (2002), pp. 556–567.

[24] Meier, A., "Parametrised enumeration," 2020, habilitation thesis, Leibniz Universität Hannover, `https://doi.org/10.15488/9427`.

[25] Meier, A. and C. Reinbold, *Enumeration complexity of poor man's propositional dependence logic*, in: *FoIKS*, Lecture Notes in Computer Science **10833** (2018), pp. 303–321.

[26] Möhle, S., R. Sebastiani and A. Biere, *On enumerating short projected models*, CoRR **abs/2110.12924** (2021).

[27] Papadimitriou, C. H., "Computational complexity," Academic Internet Publ., 2007.

[28] Pippenger, N., "Theories of computability," Cambridge University Press, 1997.

[29] Pnueli, A., *The temporal logic of programs*, in: *FOCS* (1977), pp. 46–57.

[30] Pnueli, A. and A. Zaks, *On the merits of temporal testers*, in: *25 Years of Model Checking*, Lecture Notes in Computer Science **5000** (2008), pp. 172–195.

[31] Schnoebelen, P., *The complexity of temporal logic model checking*, in: *Advances in Modal Logic* (2002), pp. 393–436.

[32] Schwarick, M., M. Heiner and C. Rohr, *MARCIE - model checking and reachability analysis done efficiently*, in: *QEST* (2011), pp. 91–100.

[33] Strozecki, Y., *Enumeration complexity*, Bull. EATCS **129** (2019).

[34] Sullivan, A., D. Marinov and S. Khurshid, *Solution enumeration abstraction: A modeling idiom to enhance a lightweight formal method*, in: *ICFEM*, Lecture Notes in Computer Science **11852** (2019), pp. 336–352.

[35] Vardi, M., *Why is modal logic so robustly decidable?*, Descriptive Complexity and Finite Models DIMACS Series in Discrete Mathematics and Theoretical Computer Science (1997), p. 149–183.