Concurrency Makes Simple Theories Hard

Stefan Göller[1] and Anthony Widjaja Lin[2]

1 **Fachbereich Informatik, University of Bremen, Germany**
2 **Department of Computer Science, Oxford University, United Kingdom**

Abstract

A standard way of building concurrent systems is by composing several individual processes by product operators. We show that even the simplest notion of product operators (i.e. asynchronous products) suffices to increase the complexity of model checking simple logics like Hennessy-Milner (HM) logic and its extension with the reachability operator (EF-logic) from PSPACE to nonelementary. In particular, this nonelementary jump happens for EF-logic when we consider individual processes represented by pushdown systems (indeed, even with only one control state). Using this result, we prove nonelementary lower bounds on the size of formula decompositions provided by Feferman-Vaught (de)compositional methods for HM and EF logics, which reduce theories of asynchronous products to theories of the components. Finally, we show that the same nonelementary lower bounds also hold when we consider the relativization of such compositional methods to finite systems.

1998 ACM Subject Classification F.4.1 Mathematical Logic, F.4.2 Grammars and Other Rewriting Systems

Keywords and phrases Modal Logic, Model Checking, Asynchronous Product, Pushdown Systems, Feferman-Vaught, Nonelementary lower bounds

Digital Object Identifier 10.4230/LIPIcs.STACS.2012.148

1 Introduction

Concurrent systems are systems which consist of multiple processes that are simultaneously executed and possibly interacting with each other. A standard way of designing concurrent systems is to compose together several individual processes by taking some "product" operators. Various product operators have been introduced in concurrency theory and verification ranging from synchronized products (the strongest form of products) to asynchronous products (the weakest form of products). From the point of view of system design, synchronized products are the most suitable form of compositional operators. Unfortunately, from the point of view of system verification, they are known to be too powerful. For example, while reachability is NL-complete for finite transition systems, it becomes PSPACE-complete when the same problem is considered over synchronized products of finite transition systems (a.k.a. communicating finite-state machines). In the case of infinite-state systems, we see a more drastic change: while reachability is decidable in polynomial time for pushdown systems (PDS), the same problem becomes undecidable when considered over synchronized products of two PDS (note: these subsume Minsky's counter machines).

In order to circumvent the problem of high complexity and undecidability in verifying concurrent systems composed from individual processes via synchronized products, various weaker notions of products were introduced. Apart from asynchronous products which prohibit the processes to communicate, stronger product operators were introduced by restricting the types of synchronization that are allowed among the processes. Several such restrictions include bounded context switches [14], and finite synchronization [18]. These

29th Symposium on Theoretical Aspects of Computer Science (STACS'12).
Editors: Christoph Dürr, Thomas Wilke; pp. 148–159

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany


restricted product operators can serve as good underapproximations of synchronized products. For example, a recent study of concurrency bugs conducted by the authors of [11] reveal that many real-world concurrency bugs can be detected within a small number of context switches. In addition, such restrictions also lead to decidability or lower computational complexity in model checking. For example, checking reachability over communicating finite-state machines and communicating pushdown systems with bounded context switches are both NP-complete [14].

When we consider logic model checking, the situation is not as simple. Asynchronous products do not make model checking easier than synchronized products when we use logics like LTL and CTL (and, in fact, even their restrictions to LTL(F,X) and the logic EG). Intuitively, the reason is that synchronization is easily embedded in such logics. Consequently, reachability of 2-stack pushdown systems, which is well-known to be undecidable, easily reduces to model checking any of aforementioned logics over asynchronous products of two PDS. In contrast, the situation is substantially better when we consider simpler logics like Hennessy-Milner (HM) Logic and its extension with the reachability operator (i.e. EF-logic). In fact, powerful *(Feferman-Vaught) compositional methods* (e.g. [12, 15, 18]), which reduce model checking of product structures to model checking of their components, can be used for obtaining decidability or better upper complexity bounds of model checking HM-logic and EF-logic. We now state the most basic form of such compositional methods from [15].

▶ **Theorem 1** ([15]). *For each HM/EF formula φ over the action labels $\mathbb{A} = \mathbb{A}_1 \cup \ldots \cup \mathbb{A}_k$, for nonempty disjoint sets $\mathbb{A}_1, \ldots, \mathbb{A}_k$, one can compute k finite sets of HM/EF formulas $\{\psi_i^1\}_{i \in I_1}, \ldots, \{\psi_i^k\}_{i \in I_k}$ over $\mathbb{A}_1, \ldots, \mathbb{A}_k$ respectively, and a positive boolean formula (i.e. no negations) β with variables $\{x_i^1\}_{i \in I_1}, \ldots, \{x_i^k\}_{i \in I_k}$ such that for* all *transition systems $\mathcal{T}_1, \ldots, \mathcal{T}_k$ with initial states $s_1, \ldots, s_k$ we have $(\prod_{i=1}^k \mathcal{T}_i, \bar{s}) \models \varphi$ if and only if $\beta[\mu]$ is true, where $\bar{s} = (s_1, \ldots, s_k)$ and μ assigns the variables of β as follows: $\mu(x_i^j) = 1$ if and only if $(\mathcal{T}_j, s_j) \models \psi_i^j$.*

Actually, a stronger version of Theorem 1 was proven in [15] (e.g. with atomic propositions). In the statement of Theorem 1, the k sets of formulas and the positive boolean formula β are referred to as the *decomposition* of φ. To give some concrete illustrations of the power of this compositional theorem, Theorem 1 can be used to show that model-checking *fixed* EF formulas (i.e. the complexity is only measured the size of the system) is: NL-complete for asynchronous products of finite systems (c.f. PSPACE-completeness of communicating finite-state systems), PSPACE-complete for asynchronous products of PDS [16], and P-complete for asynchronous products of Basic Process Algebras (equivalently, one-state PDS).

Despite the aforementioned usefulness of Feferman-Vaught compositional methods, the technique yields algorithms with nonelementary complexity in the size of the formula (see [15]), which is not desirable from both theoretical and practical viewpoints. In fact, it was recently shown that when we consider stronger logics like first-order logic, where Feferman-Vaught compositional methods are also available (e.g. see [12]), this nonelementary complexity is unavoidable [6]. It is natural to ask whether this nonelementary complexity is avoidable when we consider simpler logics like HM-logic or EF-logic. In fact, this open question has been posed in the literature (e.g. [7, 15]).

This open question actually brings us to a more fundamental open question: how do asynchronous products affect the complexity of model checking of HM-logic and EF-logic? This open question has manifested itself in the literature in various concrete forms. As an example, take the result that model checking EF-logic over pushdown systems is PSPACE-complete [20]. Over asynchronous products of *two* pushdown systems, the best algorithm for model checking EF-logic runs in nonelementary time [16]. In fact, the same nonelementary

gap is currently present for asynchronous products of two BPAs. Failing to answer this open question is also the reason for the existing nonelementary complexity gaps for several verification problems for *PA-processes* [13], which are an extension of asynchronous products of BPAs with process creations.

Contributions. In this paper, we provide answers to the above open questions. A main contribution of this paper is to show that, for each integer $k > 0$, there exists an asynchronous product of two BPAs whose EF-logic theory requires k-fold exponential time to solve. This means that model checking EF-logic over the class of asynchronous products of two BPAs requires nonelementary time, which is in stark contrast to PSPACE-completeness of EF model checking over BPAs [20]. As an upshot of our result, it follows that model checking EF-logic over PA-processes requires nonelementary time, which solves the open question posed by R. Mayr [13].

We also show that similar results hold for HM-logic. More precisely, we prove that for each integer $k > 0$ there exists an asynchronous product of two *prefix-recognizable systems* (an extension of BPAs introduced by Caucal [5] by allowing infinitely many rewrite rules compactly represented by regular languages) whose HM-logic theory requires k-fold exponential time to solve. This means that model checking HM-logic over the class of asynchronous products of two prefix-recognizable systems requires nonelementary time, which is in stark contrast to PSPACE-completeness of HM-logic model checking over prefix-recognizable systems (which easily follows[1] from the result of [20]).

An important corollary of our two aforementioned results is that there is no elementary algorithm for computing decompositions of formulas in HM-logic and EF-logic in the sense of Theorem 1. We go one step further to show that no decompositions of formulas in HM-logic and EF-logic of elementary size even exist in general. In other words, both descriptional and computational complexity of compositional methods of HM-logic and EF-logic in the sense of Theorem 1 are inherently nonelementary. Incidentally, this also entails the same nonelementary lower bounds for compositional methods provided in [7] since they generalize Theorem 1.

So far, our nonelementary lower bounds for compositional methods for HM-logic and EF-logic require the use of infinite-state systems. This still leaves the possibility that Theorem 1 could hold when we restrict the transition systems under consideration to be finite-state. Questions of this form are of particular interests in finite model theory (e.g. see [10]) and in verification of finite-state systems. We show, however, that the same nonelementary lower bounds relativize to the class of asynchronous products of finite systems. Whether the nonelementary lower bounds hold when relativized to the class of asynchronous products of *finite trees* is left for future work.

2 Preliminaries

General: By $\mathbb{N} = \{0, 1, \ldots\}$ we denote the set of *nonnegative integers*. For each $i, j \in \mathbb{N}$, we define the interval $[i, j] = \{i, i+1, \ldots, j\}$. Let $f : \mathbb{N} \to \mathbb{N}$ be a function. We write $f(n) = \mathsf{poly}(n)$ if there is some polynomial $p(n)$ such that $f(n) \leq p(n)$ for all $n \in \mathbb{N}$. We write $f(n) = \exp(n)$ if there is some polynomial $p(n)$ such that $f(n) \leq 2^{p(n)}$ for all $n \in \mathbb{N}$. We define the standard *Tower function* $\mathsf{Tower} : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ inductively as $\mathsf{Tower}(0, n) = n$ and $\mathsf{Tower}(k, n) = 2^{\mathsf{Tower}(k-1,n)}$, for each $k > 0$ and each $n \in \mathbb{N}$.

[1] On the same note, even μ-calculus over prefix-recognizable systems is only EXP-complete [9]

Automata: A *deterministic finite automaton (DFA)* is a tuple $\mathcal{A} = (Q, \Sigma, q_0, \delta, F)$, where Q is a finite set of *states*, Σ is a finite *alphabet*, $q_0 \in Q$ is the *initial state*, $\delta : Q \times \Sigma \to Q$ is the *transition function*, and $F \subseteq Q$ is the set of *final states*. By $L(\mathcal{A}) = \{w \in \Sigma^* \mid \mathcal{A} \text{ accepts } w\}$ we denote the *language* of $\mathcal{A}$. The *size* of $\mathcal{A}$ is defined as $|\mathcal{A}| = |Q|$.

Systems: Let us fix a countable set of action labels Act. A *transition system* is tuple $\mathcal{T} = (S, \mathbb{A}, \{\xrightarrow{a} \mid a \in \mathbb{A}\})$, where S is a set of *states*, $\mathbb{A} \subseteq \mathsf{Act}$ is a finite set of action labels, and where $\xrightarrow{a} \subseteq S \times S$ is a set of *transitions* for each $a \in \mathbb{A}$. We say $\mathcal{T}$ is *finite* if S is finite. A *pointed transition system* is a pair $(\mathcal{T}, s)$, where $\mathcal{T}$ is a transition system and s is state of $\mathcal{T}$. We write $s \xrightarrow{a} t$ to abbreviate $(s,t) \in \xrightarrow{a}$. We apply similar abbreviations for other binary relations over S. For each $\Gamma \subseteq \mathbb{A}$, we define $\xrightarrow{\Gamma} = \bigcup_{a \in \Gamma} \xrightarrow{a}$.

Given $k \geq 1$ transition systems $\mathcal{T}_1 = (S_1, \mathbb{A}_1, \{\xrightarrow{a}_1 \mid a \in \mathbb{A}_1\}), \ldots, \mathcal{T}_k = (S_k, \mathbb{A}_k, \{\xrightarrow{a}_k \mid a \in \mathbb{A}_k\})$, where the $\mathbb{A}_i \cap \mathbb{A}_j = \emptyset$ for each $i \neq j$, we define its *asynchronous product* $\prod_{i=1}^k \mathcal{T}_i = (S, \mathbb{A}, \{\xrightarrow{a} \mid a \in \mathbb{A}\})$, where $S = \prod_{i=1}^k S_i$, $\mathbb{A} = \bigcup_{i=1}^k \mathbb{A}_i$, and where for each $a \in \mathbb{A}$ we have $(s_1, \ldots, s_k) \xrightarrow{a} (s'_1, \ldots, s'_k)$ if and only if $s_i \xrightarrow{a}_i s'_i$ for some $i \in [1,k]$ with $a \in \mathbb{A}_i$ and $s_j = s'_j$ for each $j \in [1,k] \setminus \{i\}$.

Logic: Formulas of EF-logic over a finite set $\mathbb{A} \subseteq \mathsf{Act}$ of labels are given by the following grammar $\varphi ::= \top \mid \neg\varphi \mid \varphi \wedge \varphi \mid \langle\Gamma\rangle\varphi \mid \langle\Gamma^*\rangle\varphi$, where $\Gamma \subseteq \mathbb{A}$:
We introduce the usual abbreviations $\bot = \neg\top$, $\varphi_1 \vee \varphi_2 = \neg(\neg\varphi_1 \wedge \neg\varphi_2)$, $\varphi_1 \to \varphi_2 = \neg\varphi_1 \vee \varphi_2$, $\varphi_1 \leftrightarrow \varphi_2 = (\varphi_1 \to \varphi_2) \wedge (\varphi_2 \to \varphi_1)$, $[\Gamma]\varphi = \neg\langle\Gamma\rangle\neg\varphi$, and $[\Gamma^*]\varphi = \neg\langle\Gamma^*\rangle\neg\varphi$. We also write $\langle\Gamma\rangle^n$ (resp. $[\Gamma]^n$) as an abbreviation for a sequence of n consecutive $\langle\Gamma\rangle$'s (resp. $[\Gamma]$'s). By $|\varphi|$ we denote the *size* of each EF formula $|\varphi|$ defined as usually. *Hennessy-Milner logic (*HM*)* is the syntactic fragment of EF that one obtains by forbidding formulas of the kind $\langle\Gamma^*\rangle\varphi$. Since we allow formulas of the form $\langle\Gamma^*\rangle\varphi$ for subsets Γ of the action labels (instead of only $\langle\mathbb{A}^*\rangle$), our version of EF is slightly more general than the standard definition of EF-logic. However, our results easily carry over to the standard definition of EF-logic.

For each transition system $\mathcal{T} = (S, \mathbb{A}, \{\xrightarrow{a} \mid a \in \mathbb{A}\})$ and each EF-formula φ (over $\mathbb{A}$) define the set of states $[\![\varphi]\!]_\mathcal{T} \subseteq S$ that satisfy φ by induction on the structure of φ as follows:

$$
\begin{array}{lcllcl}
[\![\top]\!]_\mathcal{T} & = & S & [\![\langle\Gamma\rangle\varphi]\!]_\mathcal{T} & = & \{s \in S \mid \exists t \in [\![\varphi]\!]_\mathcal{T} : s \xrightarrow{\Gamma} t\} \\
[\![\neg\varphi]\!]_\mathcal{T} & = & S \setminus [\![\varphi]\!]_\mathcal{T} & [\![\langle\Gamma^*\rangle\varphi]\!]_\mathcal{T} & = & \{s \in S \mid \exists t \in [\![\varphi]\!]_\mathcal{T} : s \xrightarrow{\Gamma}{}^* t\} \\
[\![\varphi_1 \wedge \varphi_2]\!]_\mathcal{T} & = & [\![\varphi_1]\!]_\mathcal{T} \cap [\![\varphi_2]\!]_\mathcal{T} & & &
\end{array}
$$

We also write $(\mathcal{T}, s) \models \varphi$ whenever $s \in [\![\varphi]\!]_\mathcal{T}$ or simply $s \models \varphi$ when $\mathcal{T}$ is clear from the context.

Infinite-state models: A *pushdown system (PDS)* is a tuple $\mathcal{P} = (\Sigma, \mathbb{A}, \Delta)$, where Σ is a finite set of *process constants*, $\mathbb{A} \subseteq \mathsf{Act}$ is a finite set of action labels and Δ is a finite set of *rewrite rules* of the form $u \mapsto_a v$, where $a \in \mathbb{A}$, $u \in \Sigma^*$ and $v \in \Sigma^*$. A *basic process algebra (BPA)* is PDA $\mathcal{P} = (\Sigma, \mathbb{A}, \Delta)$, where for each $u \mapsto_a v \in \Delta$ we have $|u| = 1$. The associated transition system $\mathcal{T}(\mathcal{P})$ is defined as $\mathcal{T}(\mathcal{P}) = (\Sigma^*, \mathbb{A}, \{\xrightarrow{a} \mid a \in \mathbb{A}\})$, where $\xrightarrow{a} = \{(uw, vw) \mid u \mapsto_a v \in \Delta, w \in \Sigma^*\}$ for each $a \in \mathbb{A}$. The *size* of a PDS is defined as $|\mathcal{P}| = |\Sigma| + |\mathbb{A}| + \sum_{u \mapsto_a v \in \Delta}(|u| + |v|)$.

3 Hardness of asynchronous product

We start by proving a nonelementary lower bound for the problem of *model checking* EF *on BPA×BPA*: given two BPAs $\mathcal{P} = (\Sigma, \mathbb{A}, \Delta)$, $\mathcal{P}' = (\Sigma', \mathbb{A}', \Delta')$ with $\mathbb{A} \cap \mathbb{A}' = \emptyset$, a pair of process constants $\langle X, X' \rangle \in \Sigma \times \Sigma'$, and an EF formula φ over $\mathbb{A} \cup \mathbb{A}'$, check whether $(\mathcal{T}(\mathcal{P}) \times \mathcal{T}(\mathcal{P}'), \langle X, X' \rangle) \models \varphi$.

▶ **Theorem 2.** *Model checking EF on BPA×BPA is nonelementary.*

We then show that this lower bound implies a nonelementary lower bound for model checking HM-logic over the class of asynchronous products of two prefix-recognizable systems (a precise definition is given below).

3.1 Proof of Theorem 2

The structure of the proof of Theorem 2 is as follows. We first show how to encode large counters as EF formulas evaluated over the class of asynchronous products of two BPAs. Such large counters are enforced by the two stacks in the two BPAs, which alternately "guess" an encoding of a counter and "check" the correctness of the encoding. As we will see later, this encoding of large counters can be used to encode memberships of $\mathsf{Tower}(k, cn)$ space-bounded Turing machines for any fixed $k > 0$.

Large counters: The following encoding of large numbers is from [19, 3]. In the following, the notations n and ℓ will range over $\mathbb{N}$. We define the alphabets $\Omega_\ell = \{0_\ell, 1_\ell\}$ and the values $\mathsf{val}(0_\ell) = 0$ and $\mathsf{val}(1_\ell) = 1$ for each $\ell \geq 0$.

A $(1, n)$-*counter* is a word from Ω_0^n. The *value* $\mathsf{val}(c)$ of some $(1, n)$-counter $c = \sigma_0 \cdots \sigma_{n-1}$ is defined as $\mathsf{val}(c) = \sum_{i=0}^{n-1} 2^i \cdot \mathsf{val}(\sigma_i) \in [0, 2^n - 1]$. So the set of values $\mathsf{val}(c)$ for $(1, n)$-counters c equals $[0, 2^n - 1] = [0, \mathsf{Tower}(1, n) - 1]$. An (ℓ, n)-*counter* with $\ell > 1$ is a word $c = c_0\sigma_0c_1\sigma_1 \dots c_m\sigma_m$, where $m = \mathsf{Tower}(\ell - 1, n) - 1$, each c_i is an $(\ell - 1, n)$-counter with $\mathsf{val}(c_i) = i$ and $\sigma_i \in \Omega_{\ell-1}$ for each $i \in [0, m]$. We define $\mathsf{val}(c) = \sum_{i=0}^{m} 2^i \cdot \mathsf{val}(\sigma_i)$. Observe that $\mathsf{val}(c) \in [0, \mathsf{Tower}(\ell, n) - 1]$ and the length of each (ℓ, n)-counter is uniquely determined by ℓ and n.

In the following, we define $\Omega'_\ell = \{0'_\ell, 1'_\ell\}$ to be a fresh copy of Ω_ℓ; moreover define $\Sigma_\ell = \bigcup_{i=0}^{\ell} \Omega_i$ and analogously $\Sigma'_\ell = \bigcup_{i=0}^{\ell} \Omega'_i$.

Definition of the two BPAs: For each integer $\ell > 0$, let us define the following simple BPAs $\mathcal{P}_\ell = (\Sigma_\ell, \mathbb{L}_\ell, \Delta_\ell)$, where

- $\mathbb{L}_\ell = \Sigma_\ell \cup \overline{\Sigma_\ell}$, where $\overline{\Sigma_\ell} = \{\overline{\sigma} \mid \sigma \in \Sigma_\ell\}$ is a dual copy of Σ_ℓ.
- $\Delta_\ell = \{\tau \mapsto_\sigma \sigma\tau \mid \sigma, \tau \in \Sigma_\ell\} \cup \{\sigma \mapsto_{\overline{\sigma}} \varepsilon \mid \sigma \in \Sigma_\ell\}$.

The transition system $\mathcal{T}(\mathcal{P}_\ell)$ has a fairly regular behavior. The set of states is Σ_ℓ^*. Executing an action $\overline{\sigma} \in \overline{\Sigma_\ell}$ from a state $u \in (\Sigma_\ell)^*$ allows to remove exactly this leftmost symbol σ from u if u begins with σ, otherwise $\overline{\sigma}$ cannot be executed from u. Dually, from every nonempty state $u \in (\Sigma_\ell)^+$ of $\mathcal{T}(\mathcal{P}_\ell)$ we can execute every action $\sigma \in \Sigma_\ell$ yielding the state σu; the only state from which the $\sigma \in \Sigma_\ell$ are not executable is the empty word ε. We define the BPA $\mathcal{P}'_\ell$ analogously to $\mathcal{P}_\ell$ but by priming every symbol. Formally, $\mathcal{P}'_\ell = (\Sigma'_\ell, \mathbb{L}'_\ell, \Delta'_\ell)$, where

- $\mathbb{L}'_\ell = \Sigma'_\ell \cup \overline{\Sigma'_\ell}$, where $\overline{\Sigma'_\ell} = \{\overline{\sigma'} \mid \sigma' \in \Sigma'_\ell\}$ is a dual copy of Σ'_ℓ.
- $\Delta'_\ell = \{\tau' \mapsto_{\sigma'} \sigma'\tau' \mid \sigma', \tau' \in \Sigma'_\ell\} \cup \{\sigma' \mapsto_{\overline{\sigma'}} \varepsilon \mid \sigma' \in \Sigma'_\ell\}$.

Note that the set of states of $\mathcal{T}(\mathcal{P}_\ell) \times \mathcal{T}(\mathcal{P}'_\ell)$ is $(\Sigma_\ell)^* \times (\Sigma'_\ell)^*$. Given a state $s = (u, u') \in (\Sigma_\ell)^* \times (\Sigma'_\ell)^*$, we call u the *left stack of* s and u' the *right stack of* s. So we treat the words u and u' as stacks with their left-most symbols being the top of the stack. Recall that every (ℓ, n)-counter is in particular a word over $\Sigma_{\ell-1}$. We extend this notion to words over $\Sigma'_{\ell-1}$ in the usual way. So each (ℓ, n)-counter will in particular be either a word over $\Sigma_{\ell-1}$ or over $\Sigma'_{\ell-1}$, depending on whether we address the left stack or the right stack. Note that if some word over Σ_k (resp. over Σ'_k) has an (ℓ, n)-counter as a prefix, then the length of this prefix is uniquely determined by ℓ and n, namely $\mathsf{Tower}(\ell - 1, n)$. An *extended* (ℓ, n)-*counter* is either a string $c\sigma$, where either $c \in \Sigma_{\ell-1}^*$ is an (ℓ, n)-counter and $\sigma \in \Omega_\ell$, or a string $c'\sigma'$, where $c' \in (\Sigma'_{\ell-1})^*$ is an (ℓ, n)-counter and $\sigma' \in \Omega'_\ell$.

Next, we define some EF formulas (with primed counterparts for the right stack) for each $\ell, n \in \mathbb{N}$:

1. $\mathsf{count}^{\sigma}_{(\ell,n)}$ for each $\sigma \in \Omega_\ell$ such that $(\mathcal{T}(\mathcal{P}_\ell) \times \mathcal{T}(\mathcal{P}'_\ell), (u, u')) \models \mathsf{count}^{\sigma}_{(\ell,n)}$ if and only if for some (ℓ, n)-counter c we have $c\sigma$ is a prefix of u.
2. $\mathsf{count}^{\sigma'}_{(\ell,n)}$ for each $\sigma' \in \Omega_\ell$ such that $(\mathcal{T}(\mathcal{P}_\ell) \times \mathcal{T}(\mathcal{P}'_\ell), (u, u')) \models \mathsf{count}^{\sigma'}_{(\ell,n)}$ if and only if for some (ℓ, n)-counter c' we have $c'\sigma'$ is a prefix of u'.
3. $\mathsf{xcount}_{(\ell,n)}$ such that $(\mathcal{T}(\mathcal{P}_\ell) \times \mathcal{T}(\mathcal{P}'_\ell), (u, u')) \models \mathsf{xcount}_{(\ell,n)}$ if and only if some extended (ℓ, n)-counter $c\sigma$ (for $\sigma \in \Omega_\ell$) is a prefix of u.
4. $\mathsf{xcount}'_{(\ell,n)}$ such that $(\mathcal{T}(\mathcal{P}_\ell) \times \mathcal{T}(\mathcal{P}'_\ell), (u, u')) \models \mathsf{xcount}'_{(\ell,n)}$ if and only if some extended (ℓ, n)-counter $c'\sigma'$ (for $\sigma' \in \Omega'_\ell$) is a prefix of u'.
5. $\mathsf{first}_{(\ell,n)}$ (resp. $\mathsf{first}'_{(\ell,n)}$) such that $(\mathcal{T}(\mathcal{P}_\ell) \times \mathcal{T}(\mathcal{P}'_\ell), (u, u')) \models \mathsf{first}_{(\ell,n)}$ (resp. $(\mathcal{T}(\mathcal{P}_\ell) \times \mathcal{T}(\mathcal{P}'_\ell), (u, u')) \models \mathsf{first}'_{(\ell,n)}$) if and only if some extended (ℓ, n)-counter $c\sigma$ (resp. $c'\sigma'$) with $\mathsf{val}(c) = 0$ (resp. $\mathsf{val}(c') = 0$) is a prefix of u (resp. u').
6. $\mathsf{last}_{(\ell,n)}$ (resp. $\mathsf{last}'_{(\ell,n)}$) such that $(\mathcal{T}(\mathcal{P}_\ell) \times \mathcal{T}(\mathcal{P}'_\ell), (u, u')) \models \mathsf{last}_{(\ell,n)}$ (resp. $(\mathcal{T}(\mathcal{P}_\ell) \times \mathcal{T}(\mathcal{P}'_\ell), (u, u')) \models \mathsf{last}'_{(\ell,n)}$) if and only if some extended (ℓ, n)-counter $c\sigma$ (resp. $c'\sigma'$) with $\mathsf{val}(c) = \mathsf{Tower}(\ell, n) - 1$ (resp. $\mathsf{val}(c') = \mathsf{Tower}(\ell, n) - 1$) is a prefix of u (resp. u').
7. $\mathsf{eq}_{(\ell,n)}$ such that $(\mathcal{T}(\mathcal{P}_\ell) \times \mathcal{T}(\mathcal{P}'_\ell), (u, u')) \models \mathsf{eq}_{(\ell,n)}$ if and only if there exist extended (ℓ, n)-counters $c\sigma \in (\Sigma_{\ell-1})^*\Omega_\ell$ and $c'\sigma' \in (\Sigma'_{\ell-1})^*\Omega'_\ell$ such that (i) $c\sigma$ is a prefix of u, (ii) $c'\sigma'$ is a prefix of u', and (iii) $\mathsf{val}(c) = \mathsf{val}(c')$.
8. $\mathsf{inc}_{(\ell,n)}$ (resp. $\mathsf{inc}'_{(\ell,n)}$) such that $(\mathcal{T}(\mathcal{P}_\ell) \times \mathcal{T}(\mathcal{P}'_\ell), (u, u')) \models \mathsf{inc}_{(\ell,n)}$ (resp. $(\mathcal{T}(\mathcal{P}_\ell) \times \mathcal{T}(\mathcal{P}'_\ell), (u, u')) \models \mathsf{inc}'_{(\ell,n)}$) if and only if there exist extended (ℓ, n)-counters $c\sigma \in (\Sigma_{\ell-1})^*\Omega_\ell$ and $c'\sigma' \in (\Sigma'_{\ell-1})^*\Omega'_\ell$ such that (i) $c\sigma$ is a prefix of u, (ii) $c'\sigma'$ is a prefix of u', and (iii) $\mathsf{val}(c) + 1 = \mathsf{val}(c')$ (resp. $\mathsf{val}(c') + 1 = \mathsf{val}(c)$).
9. $\mathsf{succ}_{(\ell,n)}$ (resp. $\mathsf{succ}'_{(\ell,n)}$) such that $(\mathcal{T}(\mathcal{P}_\ell) \times \mathcal{T}(\mathcal{P}'_\ell), (u, u')) \models \mathsf{succ}_{(\ell,n)}$ (resp. $(\mathcal{T}(\mathcal{P}_\ell) \times \mathcal{T}(\mathcal{P}'_\ell), (u, u')) \models \mathsf{succ}'_{(\ell,n)}$) if and only if there are extended (ℓ, n)-counters $c_1\sigma_1$ and $c_2\sigma_2$ (resp. $c'_1\sigma'_1$ and $c'_2\sigma'_2$) with $\sigma_1, \sigma_2 \in \Omega_\ell$ (resp. $\sigma'_1, \sigma'_2 \in \Omega'_\ell$) such that $c_1\sigma_1c_2\sigma_2$ is a prefix of u and $\mathsf{val}(c_1) + 1 = \mathsf{val}(c_2)$ (resp. $\mathsf{val}(c'_1) + 1 = \mathsf{val}(c'_2)$).

The formulas that we will define will be exponential in ℓ and polynomial in n (represented in unary). This definition will be given by induction on ℓ. We will start with the following simple observations: $\mathsf{xcount}_{(\ell,n)} = \bigvee_{\sigma\in\Omega_\ell} \mathsf{count}^{\sigma}_{(\ell,n)}$, and $\mathsf{xcount}'_{(\ell,n)} = \bigvee_{\sigma'\in\Omega'_\ell} \mathsf{count}^{\sigma'}_{(\ell,n)}$. We will now construct several formulas φ that we evaluate on $\mathcal{T}(\mathcal{P}_\ell) \times \mathcal{T}(\mathcal{P}'_\ell)$ expressing properties of the *left stack*. Without making them explicit, we can construct corresponding analogs φ' expressing the respective property on the *right stack*.

Let us proceed by defining the above formulas for the case of $\ell = 1$. We define $\mathsf{count}^{\sigma}_{(1,n)}$ and $\mathsf{count}^{\sigma'}_{(1,n)}$ as $\mathsf{count}^{\sigma}_{(1,n)} = \langle\overline{\Omega_0}\rangle^n \langle\overline{\sigma}\rangle\top$ and $\mathsf{count}^{\sigma'}_{(1,n)} = \left\langle\overline{\Omega'_0}\right\rangle^n \langle\overline{\sigma'}\rangle\top$. We put $\mathsf{first}_{(1,n)} = \langle\overline{0_0}\rangle^n\langle\overline{\Omega_1}\rangle\top$. The definition of $\mathsf{last}_{(1,n)}$ is analogous. We also define $\mathsf{eq}_{(1,n)}$ as $\mathsf{xcount}_{(1,n)} \wedge \mathsf{xcount}'_{(1,n)} \wedge \bigwedge_{i=0}^{n-1} \langle\overline{\Omega_0}\rangle^i\langle\overline{\Omega'_0}\rangle^i \wedge \bigwedge_{\sigma\in\Omega_0}(\langle\overline{\sigma}\rangle\top \leftrightarrow \langle\overline{\sigma'}\rangle\top)$. The definitions of $\mathsf{inc}_{(1,n)}$ and $\mathsf{succ}_{(1,n)}$ are analogous.

Let us now proceed to the case of $\ell > 1$. We start by defining the formula $\mathsf{count}^{\sigma}_{(\ell,n)}$ for each $\sigma \in \Omega_\ell$. We will achieve this, by making use of the formulas $\mathsf{first}_{(\ell-1,n)}$, $\mathsf{last}_{(j,n)}$ with $j \in [1, \ell-1]$, $\mathsf{xcount}_{(\ell-1,n)}$, and $\mathsf{succ}_{(\ell-1,n)}$. The first two conjuncts of the definition of $\mathsf{count}^{\sigma}_{(\ell,n)}$ are self-explanatory,

$$\mathsf{count}^{\sigma}_{(\ell,n)} \;=\; \mathsf{first}_{(\ell-1,n)} \wedge \left[\overline{\Sigma_{\ell-1}}^*\right] \left(\mathsf{xcount}_{(\ell-1,n)} \rightarrow \left(\mathsf{last}_{(\ell-1,n)} \vee \mathsf{succ}_{(\ell-1,n)}\right)\right) \wedge \mathsf{add}^{\sigma},$$

whereas the formula add^σ will express that the symbol σ follows after the top-most (ℓ, n)-counter. Formally we put $\mathsf{add}^\sigma = \psi^\sigma_{\ell-1}$, where

$$\psi^\sigma_j = \begin{cases} \left[\overline{\Sigma_j}^*\right] \left(\mathsf{last}_{(j,n)} \to \psi^\sigma_{j-1}\right) & \text{if } j > 1 \\ \left[\overline{\Sigma_1}^*\right] \left(\mathsf{last}_{(1,n)} \to \langle \overline{1_0} \rangle^n \langle \overline{1_1} \rangle \langle \overline{1_2} \rangle \cdots \langle \overline{1_{\ell-2}} \rangle \langle \overline{\Omega_{\ell-1}} \rangle \langle \overline{\sigma} \rangle \top\right) & \text{if } j = 1. \end{cases}$$

Intuitively, the formula $\psi^\sigma_{\ell-1}$ jumps to last extended $(1, n)$-counter of the last extended $(2, n)$-counter ... of the last extended $(\ell - 1, n)$-counter and expresses that the correct sequence follows from this position.

Define $\mathsf{first}_{(\ell,n)}$ as $\mathsf{first}_{(\ell,n)} = \mathsf{xcount}_{(\ell,n)} \wedge \left[\overline{\Sigma_{\ell-1}}^*\right] \left(\langle \overline{\Omega_{\ell-1}} \rangle \top \to \langle \overline{0_{\ell-1}} \rangle \top\right)$, similarly we define $\mathsf{last}_{(\ell,n)}$. We set $\mathsf{eq}_{(\ell,n)}$ as the conjunction of $\mathsf{xcount}_{(\ell,n)} \wedge \mathsf{xcount}'_{(\ell,n)}$ and

$$\left[\overline{\Sigma_{\ell-1}}^*\right] \left(\mathsf{xcount}_{(\ell-1,n)} \to \left(\left\langle \overline{\Sigma'_{\ell-1}}^* \right\rangle \left(\mathsf{eq}_{(\ell-1,n)} \wedge \bigwedge_{\sigma \in \Omega_{\ell-1}} (\langle \overline{\Sigma^*_{\ell-2}} \rangle \langle \overline{\sigma} \rangle \top \leftrightarrow \langle \overline{\Sigma'^*_{\ell-2}} \rangle \langle \overline{\sigma'} \rangle \top)\right)\right)\right).$$

Let us give some intuition on the formulas $\mathsf{eq}_{(\ell,n)}$ for each $\ell \in [2, k]$: Whenever we pop from the left stack some string from $(\Sigma_{\ell-1})^*$ until on top of the left stack there is some extended $(\ell - 1, n)$-counter $c\sigma$, one can remove from the right stack a string from $(\Sigma'_{\ell-1})^*$ yielding an extended $(\ell - 1, n)$ counter $c'\tau'$ on top of the right stack such that $\mathsf{val}(c) = \mathsf{val}(c')$ and moreover $\sigma = \tau$ holds.

In analogy to $\mathsf{eq}_{(\ell,n)}$ one can define the formula $\mathsf{inc}_{(\ell,n)}$. Finally, let us define $\mathsf{succ}_{(\ell,n)}$. We put

$$\mathsf{succ}_{(\ell,n)} \;=\; \langle \Sigma'_\ell \rangle \left\langle (\Sigma'_{\ell-1})^* \right\rangle \left(\mathsf{eq}_{(\ell,n)} \wedge \left\langle \overline{\Sigma_{\ell-1}}^* \right\rangle \langle \overline{\Sigma_\ell} \rangle \, \mathsf{inc}'_{(\ell,n)}\right)$$

Intuitively, we the formula $\mathsf{succ}_{(\ell,n)}$ pushes onto the right stack some string that it checks to be a copy of the topmost extended (ℓ, n)-counter of the left stack via $\mathsf{eq}_{(\ell,n)}$, then pops the topmost extended (ℓ, n)-counter of the left stack and then invokes the formula $\mathsf{inc}'_{(\ell,n)}$.

It is easy to see that the formulas given above express the desired properties. Furthermore, we note that the size of each formula is exponential in ℓ and polynomial in n.

By using standard arguments (e.g. see the proof of PSPACE-hardness of EF model checking over pushdown systems in [2]), we can now complete the proof of Theorem 2. In the following, we shall only provide a sketch. For each integer $k > 0$, there exists a fixed Turing machine $\mathcal{M}$ operating with $\mathsf{Tower}(k - 1, cn)$ space (for a constant c) whose membership problem is complete for $\mathsf{SPACE}(\mathsf{Tower}(k - 1, \mathsf{poly}(n)))$. Each such membership problem can easily be reduced to EF model checking over the class of asynchronous products of two BPAs in polynomial time as follows. For an input word w of $\mathcal{M}$ of length n one can construct formulas $\mathsf{xcount}_{(k,cn)}$ and the pair of BPAs $\mathcal{P}_{k+1}$ and $\mathcal{P}'_{k+1}$ in time $\mathsf{poly}(n)$. Each computation of $\mathcal{M}$ can be viewed as a sequence of configurations (each being a (k, cn)-counters), which when considered together is also a $(k + 1, cn)$-counter, satisfying the transition conditions of $\mathcal{M}$ (e.g. two consecutive configurations respect the transition function of $\mathcal{M}$). One can express the computation of $\mathcal{M}$ on w by an EF formula of the kind $\langle \Sigma^*_{k+1} \rangle \varphi$, where $\langle \Sigma^*_{k+1} \rangle$ aims at pushing a sequence onto the left stack and where φ expresses that this sequence expresses the desired properties.

We will make use of the following lemma in Section 4.

▶ **Lemma 3.** *Every DFA accepting the regular language*

$$L_{\ell,n} \;=\; \{u \in \Sigma^*_\ell \mid \exists u' \in (\Sigma'_\ell)^* : (\mathcal{T}(\mathcal{P}_\ell) \times \mathcal{T}(\mathcal{P}'_\ell), (u, u')) \models \mathsf{xcount}_{(\ell,n)}\}$$

has at least $\mathsf{Tower}(\ell - 1, n) + 1$ *states.*

Proof. Recall that every extended (ℓ, n)-counter has length exactly $\mathsf{Tower}(\ell - 1, n) + 1$. We have $L_{\ell,n} = \{c\sigma w \mid w \in \Sigma_\ell^*, c\sigma \text{ is some extended } (\ell, n)\text{-counter}\}$. The corollary now follows from the following simple observation: Every DFA $\mathcal{A}$ over some alphabet Σ with $L(\mathcal{A}) = U \cdot \Sigma^*$ for some $\emptyset \subsetneq U \subseteq \Sigma^m$ has at least m states. ◀

3.2 Lower bounds for HM-logic

We conclude this section by showing how Theorem 2 implies a nonelementary lower bound for model checking HM on the asynchronous product of two prefix-recognizable systems. A *prefix-recognizable system* is a tuple $\mathcal{R} = (\Sigma, \mathbb{A}, \Delta)$, where Σ is finite set of process constants, $\mathbb{A} \subseteq \mathsf{Act}$ is a finite set of action labels and Δ is a finite set of rewrite rules of the form $U \mapsto_a V$, where $a \in \mathbb{A}$, and where $U, V \subseteq \Sigma^*$ are regular languages given as DFAs, say. The associated transition system is $\mathcal{T}(\mathcal{R}) = (\Sigma^*, \mathbb{A}, \{\xrightarrow{a} \mid a \in \mathbb{A}\})$, where $\xrightarrow{a} = \{(uw, vw) \mid u \in U, v \in V, w \in \Sigma^* \text{ for some rule } U \mapsto_a V \in \Delta\}$ for each $a \in \mathbb{A}$.

Remark: One can construct from a given pair of BPAs $\mathcal{P} = (\Sigma, \mathbb{A}, \Delta)$ and $\mathcal{P}' = (\Sigma', \mathbb{A}', \Delta')$ and a given EF formula φ over $\mathbb{A} \cup \mathbb{A}'$ a pair of prefix-recognizable systems $\mathcal{R} = (\Sigma, \mathbb{A}, \Delta_{\mathcal{R}})$ and $\mathcal{R}' = (\Sigma', \mathbb{A}', \Delta'_{\mathcal{R}})$ and some HM formula $\widetilde{\varphi}$ such that $[\![\varphi]\!]_{\mathcal{T}(\mathcal{P})\times\mathcal{T}(\mathcal{P})} = [\![\widetilde{\varphi}]\!]_{\mathcal{T}(\mathcal{R})\times\mathcal{T}(\mathcal{R}')}$ as follows: By [4] one can compute for each $\Gamma \subseteq \mathbb{A}$ (analogously for each $\Gamma' \subseteq \mathbb{A}'$) a pair regular languages U_Γ and V_Γ (resp. $U_{\Gamma'}$ and $V_{\Gamma'}$) each accepted by DFAs of at most exponential size such that the relation $\xrightarrow{\Gamma}^*$ over $\Sigma^* \times \Sigma^*$ (resp. $\xrightarrow{\Gamma'}^*$ over $(\Sigma')^* \times (\Sigma')^*$) is exactly $\mathcal{R}(\widetilde{\Gamma}) = \{(uw, vw) \mid u \in U_\Gamma, v \in V_\Gamma, w \in \Sigma^*\}$ (resp. $\mathcal{R}'(\widetilde{\Gamma'}) = \{(uw, vw) \mid u \in U_{\Gamma'}, v \in V_{\Gamma'}, w \in (\Sigma')^*\}$). The latter is even shown for PDAs in [4]. Hence we can define the HM formula $\widetilde{\varphi}$ to emerge from φ by replacing each occurence of $\langle\Gamma^*\rangle$ by $\langle\widetilde{\Gamma}\rangle$ and each occurrence of $\langle(\Gamma')^*\rangle$ by $\langle\widetilde{\Gamma'}\rangle$.

Theorem 2 and the previous remark immediately imply the following corollary.

▶ **Corollary 4.** *Model checking* HM *on the asynchronous product of two prefix-recognizable systems is nonelementary.*

We remark that model checking HM on a single prefix-recognizable system is only PSPACE-complete; the upper bound can be shown via reduction to EF model checking pushdown systems, which is in PSPACE by [20].

4 Lower bounds for compositional methods for HM and EF logics

We start by proving nonelementary lower bounds for Feferman-Vaught type of compositional methods for HM and EF logics (i.e. Theorem 1) already over the the class of asynchronous products of *two* transition systems. In Section 4.2 we will then show how our lower bounds can be relativized to the class of all asynchronous products of *two finite* transition systems.

Let us briefly recall decompositions following Theorem 1 for EF logic of asynchronous products of two transition systems. Analogously HM can be dealt with. A *decomposition* of the asynchronous product of two transition systems, the first component being defined over action labels $\mathbb{A}$ and the second one over $\mathbb{A}'$ (we assume that any two such sets $\mathbb{A}$ and $\mathbb{A}'$ are non-empty and disjoint for the rest of this section) is a triple $\mathcal{D} = (\Psi, \Psi', \beta)$, where $\Psi = \{\psi_i\}_{i\in I}$ and $\Psi' = \{\psi'_j\}_{j\in J}$ for index sets I and J, where β is a positive boolean formula with variables ranging over $\{x_i\}_{i\in I} \cup \{x'_j\}_{j\in J}$, each $\psi \in \Psi$ (resp. each $\psi' \in \Psi'$) is an EF formula that is interpreted on the first (resp. second) component, i.e. over $\mathbb{A}$ (resp. $\mathbb{A}'$).

Recall that such a decomposition has the property that for every pointed transition system $(\mathcal{T}, s)$ over $\mathbb{A}$ and every pointed transition system $(\mathcal{T}', s')$ over $\mathbb{A}'$ and every EF formula φ over $\mathbb{A} \cup \mathbb{A}$ we have $((\mathcal{T} \times \mathcal{T}'), (s, s')) \models \varphi$ if and only if $\beta[\mu]$ is true, where $\mu(x_i) = 1$ if and only if $(\mathcal{T}, s) \models \psi_i$ and where $\mu(x'_j) = 1$ if and only if $(\mathcal{T}', s') \models \psi'_j$. As expected, the *size* of such a decomposition is defined as $|\mathcal{D}| = \sum_{\psi \in \Psi} |\psi| + \sum_{\psi' \in \Psi'} |\psi'| + |\beta|$.

▶ **Theorem 5.** *The size of decompositions for* EF *(resp.* HM*) formulas in the sense of Theorem 1 cannot be bounded by an elementary function. More precisely, there is a family of* EF *(resp.* HM*) formulas* $\{\varphi_\ell \mid \ell \geq 1\}$ *where* φ_ℓ *is defined over some action labels* $\mathbb{A}_\ell \cup \mathbb{A}'_\ell$*, such that* $|\varphi_\ell| = \exp(\ell)$*, and such that for every elementary function* $f : \mathbb{N} \to \mathbb{N}$ *there is some* $h \in \mathbb{N}$ *such that every decomposition* $\mathcal{D}$ *for* φ_h *on the class of all asynchronous products of two transition systems over, respectively,* $\mathbb{A}_h$ *and* $\mathbb{A}'_h$ *satisfies* $|\mathcal{D}| > f(h)$*.*

4.1 Proof of Theorem 5

The proof idea for Theorem 5 for the case of EF-logic is as follows (we will remark how to adapt it for HM-logic later). We consider the sequence of pairs of BPAs $\{(\mathcal{P}_\ell, \mathcal{P}'_\ell)\}_{\ell \geq 1}$ defined in the previous section, where the set of states of $\mathcal{T}(\mathcal{P}_\ell)$ (resp. $\mathcal{T}(\mathcal{P}'_\ell)$) is Σ_ℓ^* (resp. $(\Sigma'_\ell)^*$). We will show that if a small (i.e. of elementary size) decomposition for EF-formulas exists in general, then there is a family of DFAs $\mathcal{A}_\ell$ of size elementary in ℓ with $L(\mathcal{A}_\ell) = L_{\ell,\ell}$ for each ℓ, clearly contradicting Lemma 3. To this end, we invoke the result from [2] about the sizes of automata expressing the sets of configurations of BPAs satisfying EF formulas combined with standard constructions from automatic structures.

We first recall the following proposition from [2] about the size of DFAs representing the set of configurations of BPAs satisfying EF formulas.

▶ Proposition 6 ([2]). Given an EF formula φ and a BPA $\mathcal{P} = (\Sigma, \mathbb{A}, \Delta)$, there exists a DFA $\mathcal{A}_\varphi$ of size double exponential in $|\mathcal{P}| + |\varphi|$ with $L(\mathcal{A}_\varphi) = [\![\varphi]\!]_{\mathcal{T}(\mathcal{P})}$, i.e. $\mathcal{A}_\varphi$ accepts the set of states u of $\mathcal{T}(\mathcal{P})$ with $(\mathcal{T}(\mathcal{P}), u) \models \varphi$.

Actually, in [2], the authors construct alternating finite automata with polynomially many states, which can be translated to DFAs of double exponential size (e.g. see [17]).

Define $\{\varphi_\ell \mid \ell \geq 1\}$ as $\varphi_\ell = \mathsf{xcount}_{(\ell,\ell)}$ over the action labels $\mathbb{A}_\ell = \mathbb{L}_\ell$ and $\mathbb{A}'_\ell = \mathbb{L}'_\ell$, where recall that $\mathbb{L}_\ell$ (resp. $\mathbb{L}'_\ell$) are the action labels of the BPA $\mathcal{P}_\ell$ (resp. $\mathcal{P}'_\ell$) defined in the previous section.

To prove Theorem 5, assume to the contrary that the there exist decompositions for EF formulas φ whose sizes can be bounded from above by an elementary function, say by $\mathsf{Tower}(r, |\varphi|)$ for some *fixed* $r \in \mathbb{N}$. Let $h \in \mathbb{N}$ be a sufficiently large number for the following arguments to work. Let us fix a smallest possible decomposition $\mathcal{D} = (\Psi, \Psi', \beta)$ for the EF formula $\varphi_h = \mathsf{xcount}_{(h,h)}$ over $\mathbb{L}_h \cup \mathbb{L}'_h$. Thus by assumption $|\mathcal{D}| \leq \mathsf{Tower}(r, |\varphi_h|)$. Let $\Psi = \{\psi_i\}_{i \in I}$ and $\Psi' = \{\psi'_j\}_{j \in J}$. Recall that each $\psi_i \in \Psi$ is an EF formula over $\mathbb{L}_h$, and each $\psi'_j \in \Psi'$ is an EF formula over $\mathbb{L}'_h$. Moreover β is a positive boolean formula over the variables $\{x_i\}_{i \in I} \cup \{x'_j\}_{j \in J}$ such that for every state $(u, u') \in (\Sigma_h)^* \times (\Sigma'_h)^*$ of $\mathcal{T}(\mathcal{P}_h) \times \mathcal{T}(\mathcal{P}'_h)$, it is the case that $(\mathcal{T}(\mathcal{P}_h) \times \mathcal{T}(\mathcal{P}'_h), (u, u')) \models \varphi_h$ if and only if $\beta[\mu]$ is true, where μ is the assignment to β where we have $\mu(x_i) = 1$ if and only if $(\mathcal{T}(\mathcal{P}_h), u) \models \psi_i$ and $\mu(x'_j) = 1$ if and only if $(\mathcal{T}(\mathcal{P}'_h), u') \models \psi'_j$.

Next, we will use Proposition 6 and the small decomposition given by the assumption to construct a DFA for the language $L_{h,h} = \{u \in \Sigma_h^* \mid \exists u \in (\Sigma'_h)^* : (\mathcal{T}(\mathcal{P}_h) \times \mathcal{T}(\mathcal{P}'_h), (u, u')) \models \varphi_h\}$ with less than $\mathsf{Tower}(h-1, h) + 1$ states, which will contradict Lemma 3. To do so, we first make the following simple observation that relates the decomposition $\mathcal{D}$ of φ_h and the formula φ_h itself.

Define the EF formula $\widetilde{\beta}$ over $\mathbb{L}_h \cup \mathbb{L}'_h$ to be obtained from the boolean formula β by replacing each variable x_i by ψ_i and each variable x'_j by ψ'_j. Then, since all formulas ψ_i and ψ'_j are also formulas over $\mathbb{L}_h \cup \mathbb{L}'_h$, the EF formula $\widetilde{\beta}$ is also a formula over $\mathbb{L}_h \cup \mathbb{L}'_h$. Moreover, it is easy to see that by assumption we have $[\![\varphi_h]\!]_{\mathcal{T}(\mathcal{P}_h)\times\mathcal{T}(\mathcal{P}'_h)} = [\![\widetilde{\beta}]\!]_{\mathcal{T}(\mathcal{P}_h)\times\mathcal{T}(\mathcal{P}'_h)}$. In fact, the latter immediately follows from the fact that

$$[\![\psi_i]\!]_{\mathcal{T}(\mathcal{P}_h)\times\mathcal{T}(\mathcal{P}'_h)} = [\![\psi_i]\!]_{\mathcal{T}(\mathcal{P}_h)} \times (\Sigma'_h)^*, \quad \text{and} \tag{1}$$

$$[\![\psi'_j]\!]_{\mathcal{T}(\mathcal{P}_h)\times\mathcal{T}(\mathcal{P}'_h)} = \Sigma_h^* \times [\![\psi'_j]\!]_{\mathcal{T}(\mathcal{P}'_h)} \tag{2}$$

which can easily be proven by induction on the structure of the formulas ψ_i and ψ'_j since no action labels of $\mathcal{P}_h$ (resp. $\mathcal{P}'_h$) occur in the action labels of ψ'_j (resp. ψ_i). Thus, the goal to obtain a contradiction will be to show that we can find a small DFA for

$$L_1(\widetilde{\beta}) = \{u \in \Sigma_h^* \mid \exists u' \in (\Sigma'_h)^* : (\mathcal{T}(\mathcal{P}_h) \times \mathcal{T}(\mathcal{P}'_h), (u, u')) \models \widetilde{\beta}\}.$$

Using Proposition 6 we obtain DFAs for $[\![\psi_i]\!]_{\mathcal{T}(\mathcal{P}_h)}$ (for each $i \in I$) and $[\![\psi'_j]\!]_{\mathcal{T}(\mathcal{P}'_h)}$ (for each $j \in J$) each of size double exponential in, respectively, $|\psi_i| + |\mathcal{P}_h|$ and $|\psi_j| + |\mathcal{P}'_h|$. To obtain a small DFA for $L_1(\widetilde{\beta})$ from these DFAs, we will now perform some simple constructions from automatic structures (e.g. see [16]). We first briefly recall the notion of (binary) automatic relations. Fix a nonempty finite alphabet Σ. A pair of words $(u, w) = (a_1 \cdots a_m, b_1 \cdots b_n) \in \Sigma^* \times \Sigma^*$ can be represented as a word $u \otimes w = c_1 \cdots c_k$ of length $k = \max(m, n)$ in the new alphabet $\Sigma_\perp \times \Sigma_\perp$, where $\Sigma_\perp = \Sigma \cup \{\perp\}$ with a "padding" symbol $\perp \notin \Sigma$, and where either $c_i = (a_i, b_i)$ if $i \leq m$ and $i \leq n$, where $c_i = (a_i, \perp)$ if $i \leq m, i > n$ or where $c_i = (\perp, b_i)$ otherwise. A (binary) relation $R \subseteq \Sigma^* \times \Sigma^*$ is said to be *automatic* if the language $\{u \otimes v \mid (u, v) \in R\} \subseteq (\Sigma_\perp \times \Sigma_\perp)^*$ can be accepted by a DFA (i.e. is regular). We also write $\pi_1(R)$ to be the projection of R to the first component, i.e., $\pi_1(R) = \{u \in \Sigma^* \mid \exists w : (u, w) \in R\}$. The following proposition is standard (e.g. see [16]):

▶ Proposition 7. Given two automatic relations R_1, R_2 accepted by DFAs $\mathcal{A}_1$ and $\mathcal{A}_2$, respectively, we have (i) the relation $R_1 \cap R_2$ can be accepted by a DFA of size at most $|\mathcal{A}_1| \cdot |\mathcal{A}_2|$, (ii) the relation $R_1 \cup R_2$ can be accepted by a DFA of size at most $|\mathcal{A}_1| \cdot |\mathcal{A}_2|$, and (iii) the language $\pi_1(R_1) \subseteq \Sigma^*$ can be accepted by a DFA of size $2^{O(|\mathcal{A}_1|)}$.

Observe now that $[\![\psi_i]\!]_{\mathcal{T}(\mathcal{P}_h)\times\mathcal{T}(\mathcal{P}'_h)}$ (for each $i \in I$) and $[\![\psi'_j]\!]_{\mathcal{T}(\mathcal{P}_h)\times\mathcal{T}(\mathcal{P}'_h)}$ (for each $j \in J$) is an automatic relation over the alphabet $\Sigma = \Sigma_h \cup \Sigma'_h$ that can be accepted by DFAs of size double exponential in, respectively, $|\psi_i| + |\mathcal{P}_h| + |\mathcal{P}'_h|$ and $|\psi'_j| + |\mathcal{P}_h| + |\mathcal{P}'_h|$ by Proposition 6. The construction of a small DFA $\mathcal{A}$ for the language $L_1(\widetilde{\beta})$ can be done in a bottom-up fashion with respect to $\widetilde{\beta}$ using Proposition 7 by firstly taking unions and intersections from the DFAs recognizing $[\![\psi_i]\!]_{\mathcal{T}(\mathcal{P}_h)\times\mathcal{T}(\mathcal{P}'_h)}$ (for each $i \in I$) and $[\![\psi'_j]\!]_{\mathcal{T}(\mathcal{P}_h)\times\mathcal{T}(\mathcal{P}'_h)}$ (for each $j \in J$), and at the end projecting to the first component. All in all, there are constants c_1, c_2 with $c_1 < c_2$ (both independent of h) such that

$$|\mathcal{A}| \leq \mathsf{Tower}(c_1, |\varphi_h| + |\mathcal{P}_h| + |\mathcal{P}'_h|) \leq \mathsf{Tower}(c_2, h). \tag{3}$$

The latter inequality follows from the fact that $|\mathcal{P}_h| + |\mathcal{P}'_h| = \mathsf{poly}(h)$ and $|\varphi_h| = \exp(h)$. On the other hand, due to $[\![\varphi_h]\!]_{\mathcal{T}(\mathcal{P}_h)\times\mathcal{T}(\mathcal{P}'_h)} = [\![\widetilde{\beta}]\!]_{\mathcal{T}(\mathcal{P}_h)\times\mathcal{T}(\mathcal{P}'_h)}$ and Lemma 3, we must have

$$|\mathcal{A}| \geq \mathsf{Tower}(h - 1, h) + 1. \tag{4}$$

It is clear that if we choose h sufficiently large, then inequalities (3) and (4) cannot hold at the same time, a contradiction.

Remark. The proof above can be easily adapted to the case of HM-logic by taking prefix-recognizable systems and the HM formulas of the form $\widetilde{\mathsf{xcount}_{(\ell,\ell)}}$ defined in the remark given at the end of previous section instead of BPAs and EF formulas of the form $\mathsf{xcount}_{(\ell,\ell)}$.

4.2 Restricting to finite transition systems

Theorem 5 gives a nonelementary lower bound for decompositions over asynchronous products of two general transition systems. This still leaves the possibility that better upper bounds might be possible when we consider only asynchronous products of *finite* transition systems, i.e., the version of Theorem 1 when transition systems under consideration are finite. The following theorem shows that this is not the case.

▶ **Theorem 8.** *The size of decompositions for* EF *(resp.* HM*) formulas in the sense of Theorem 1 cannot be bounded by an elementary function when restricted to the class of finite transition systems.*

Roughly speaking, this theorem can be proven by combining Theorem 5 and the fact that HM and EF logics satisfy "finite model property with respect to a finite set of formulas": a logic $\mathcal{L}$ is said to satisfy the *finite model property with respect to a finite set of formulas* whenever, for every finite set Ξ of $\mathcal{L}$-formulas and every pointed transition system $(\mathcal{T}, s)$ there exists a *finite* pointed transition system $(\mathcal{T}_\Xi, s_\Xi)$ such that for all $\psi \in \Xi$ we have $(\mathcal{T}, s) \models \psi$ if and only if $(\mathcal{T}_\Xi, s_\Xi) \models \psi$.

It simple to check that when restricted to logics that are closed under boolean operations the finite model property with respect to a finite set of formulas is equivalent to the the finite model property (for single formulas). To prove Theorem 8, the following technical lemma suffices.

▶ **Lemma 9.** *Let φ be an* HM *(resp.* EF*) formula over the action labels $\mathbb{A} \cup \mathbb{A}'$ for nonempty disjoint sets $\mathbb{A}$ and $\mathbb{A}'$. Then, every decomposition of φ over all asynchronous products of two finite systems over $\mathbb{A}$ and $\mathbb{A}$, respectively, is also a decomposition of φ over all asynchronous products of two general transition systems over $\mathbb{A}$ and $\mathbb{A}'$, respectively.*

Observe that Theorem 5 and the above lemma immediately imply Theorem 8. We shall give the proof of Lemma 9 for EF-logic. In fact, HM-logic can be dealt with completely analogously. We note that EF (analogously HM) has the finite model property with respect to a finite set of formulas owing to the same property for *Propositional Dynamic Logic* (of which EF-logic is a sublogic). The latter can be proven e.g. via filtration (e.g. see [8]).

Let us now proceed to the proof of Lemma 9. Let fix an arbitrary EF-formula φ. Let $\mathcal{C} = (\Phi, \Phi', \alpha)$ be a decomposition of φ over the class asynchronous products of two *finite* transition systems over $\mathbb{A}$ and $\mathbb{A}'$, respectively. Let us assume $\Phi = \{\varphi_k\}_{k\in K}$ and $\Phi' = \{\varphi'_m\}_{m\in M}$ for index sets K and M. We assume here that α is a boolean formula over the variables $\{x_k\}_{k\in K}$ and $\{x'_m\}_{m\in M}$. It is important to note here that we may only assume here that $\mathcal{C}$ is a decomposition for φ on the class of asynchronous products of two *finite* transition systems over $\mathbb{A}$ and $\mathbb{A}'$, respectively.

In addition, we apply Theorem 1 to obtain a decomposition $\mathcal{D} = (\Psi, \Psi', \beta)$ of φ over the class of asynchronous products of two *general* transition systems, where $\Psi = \{\psi_i\}_{i\in I}$ and $\Psi' = \{\psi_j\}_{j\in J}$. We assume that β is a boolean formula over the variables $\{y_i\}_{i\in I}$ and $\{y'_j\}_{j\in J}$.

Let us define the finite set of formulas $\Xi = \Phi \cup \Psi$ over $\mathbb{A}$ and $\Xi' = \Phi' \cup \Psi'$ over $\mathbb{A}'$. Let us fix an arbitrary pointed transition system $(\mathcal{T}, s)$ over $\mathbb{A}$ and an arbitrary transition system $(\mathcal{T}', s')$ over $\mathbb{A}'$. Let us moreover fix some *finite* pointed transition systems $(\mathcal{T}_\Xi, s_\Xi)$ over $\mathbb{A}$ and $(\mathcal{T}'_{\Xi'}, s'_{\Xi'})$ over $\mathbb{A}'$ witnessing the finite model property with respect to Ξ and Ξ', respectively. To prove to lemma we will show that already $\mathcal{C}$ is a decomposition, so we will show $(\mathcal{T} \times \mathcal{T}', (s, s')) \models \varphi$ if and only if α is true, if $x_k = ((\mathcal{T}, s) \models \varphi_k)$ and

$x'_m = ((\mathcal{T}', s') \models \varphi'_m)$. The latter follows from the following equivalences:

$$\begin{aligned}
(\mathcal{T} \times \mathcal{T}', (s, s')) \models \varphi \quad &\Leftrightarrow \quad \beta \text{ is true if } y_i = ((\mathcal{T}, s) \models \psi_i) \text{ and } y'_j = ((\mathcal{T}', s') \models \psi'_j).\\
&\Leftrightarrow \quad \beta \text{ is true if } y_i = ((\mathcal{T}_\Xi, s_\Xi) \models \psi_i) \text{ and } y'_j = ((\mathcal{T}'_{\Xi'}, s'_{\Xi'}) \models \psi'_j).\\
&\Leftrightarrow \quad (\mathcal{T}_\Xi \times \mathcal{T}'_{\Xi'}, (s_\Xi, s'_{\Xi'})) \models \varphi.\\
&\Leftrightarrow \quad \alpha \text{ is true if } x_k = ((\mathcal{T}_\Xi, s_\Xi) \models \varphi_k) \text{ and } x'_m = ((\mathcal{T}'_{\Xi'}, s'_{\Xi'}) \models \varphi'_m).\\
&\Leftrightarrow \quad \alpha \text{ is true if } x_k = ((\mathcal{T}, s) \models \varphi_k) \text{ and } x'_m = ((\mathcal{T}', s') \models \varphi'_m).
\end{aligned}$$

Hence, $\mathcal{C}$ is a decomposition over all (both finite and infinite) transition systems, completing the proof of Lemma 9.

Acknowledgements: Anthony Lin thanks EPSRC (EP/H026878/1) for their support.

References

1. P. Blackburn, M. de Rijke, Y. Venema. *Modal Logic*. Cambridge University Press, 2001.
2. A. Bouajjani, J. Esparza, O. Maler. Reachability Analysis of Pushdown Automata: Applications to Model-Checking. In *CONCUR'97*, p. 135–150.
3. T. Cachat, I. Walukiewicz. The Complexity of Games on Higher Order Pushdown Automata. In *arxiv* http://arxiv.org/abs/0705.0262
4. D. Caucal. On the Regular Structure of Prefix Rewriting In *CAAP'90*, p. 87–102.
5. D. Caucal. On infinite transition graphs having a decidable monadic theory. In *ICALP'96*, p. 194–205.
6. A. Dawar, M. Grohe, S. Kreutzer, N. Schweikardt. Model Theory Makes Formulas Large. In *ICALP'07*, p. 913–924.
7. I. Felscher, W. Thomas. Compositionality and Reachability with Conditions on Path Lengths. *Int. J. Found. Comput. Sci.* 20(5): 851–868 (2009)
8. D. Harel, D.Kozen, J. Tiuryn. *Dynamic Logic*. MIT Press, 2000.
9. O. Kupferman, N. Piterman, M. Y. Vardi. Model checking linear properties of prefix-recognizable systems. In *CAV'02*, p. 371–385.
10. L. Libkin. *Elements of Finite Model Theory*. Springer, 2004.
11. S. Lu, S. Park, E. Seo, Y. Zhou. Learning from mistakes: a comprehensive study on real world concurrency bug characteristics. In *ASPLOS'08*, p. 329–339.
12. J. A. Makowsky. Algorithmic uses of the Feferman-Vaught Theorem. *Ann. Pure Appl. Logic* 126(1–3): 159–213 (2004)
13. R. Mayr. *Decidability and Complexity of Model Checking Problems for Infinite-State Systems*. PhD thesis, TU München, 1998.
14. S. Qadeer, J. Rehof. Context-Bounded Model Checking of Concurrent Software. In *TACAS'05*, p. 93–107.
15. A. Rabinovich. On compositionality and its limitations. *ACM TOCL* 8(1): (2007)
16. A. W. To. Model Checking Infinite-State Systems: Generic and Specific Approaches. PhD thesis, University of Edinburgh, 2010.
17. M. Vardi. An Automata-Theoretic Approach to Linear Temporal Logic In *Banff Higher Order Workshop '95*, p. 238-266.
18. S. Wöhrle, W. Thomas. Model Checking Synchronized Products of Infinite Transition Systems. *LMCS* 3(4): (2007)
19. I. Walukiewicz. Difficult Configurations - On the Complexity of LTrL In *ICALP '98*, p. 140–151.
20. I. Walukiewicz. Model Checking CTL Properties of Pushdown Systems. In *FSTTCS'00*, p. 127–138.