A Comparison of Succinctly Represented Finite-state Systems *

Romain Brenguier[1], Stefan Göller[2], and Ocan Sankur[1]

[1] LSV, CNRS & ENS Cachan, France.
[2] Institut für Informatik, Universität Bremen, Germany

Abstract. We study the succinctness of different classes of succinctly presented finite transition systems with respect to bisimulation equivalence. Our results show that synchronized product of finite automata, hierarchical graphs, and timed automata are pairwise incomparable in this sense. We moreover study the computational complexity of deciding simulation preorder and bisimulation equivalence on these classes.

1 Introduction

In formal verification *model checking* is one of the most successful approaches; it asks to test automatically whether a given system meets a given specification. Unfortunately, model checking tools have to deal with a combinatorial blow up of the state space, commonly known as the *state explosion problem*, that can be seen as one of the biggest challenges in real-world problems. Different sources of explosion arise, for instance the number of *program variables* or *clocks*, the number of *concurrent components*, or the number of different *subroutines*, just to mention few of them. Numerous techniques to tame the state explosion problem have been introduced such as abstraction methods, partial order reduction or counterexample guided abstraction refinement.

Flip side of the coin, when modeling everyday systems that are potentially exponentially big (also called the *flattened system* or just *flat system*), it is desirable to have succinct representations for them. Three fundamental models include (i) *products of flat systems*, (ii) *timed automata* (more precisely the transitions systems evolving from the time-abstract semantics of timed automata), and (iii) *hierarchical systems*, each of them successfully being used to tame the state explosion problem in their dimension (these dimensions are pairwise orthogonal): (i) Products of flat systems allow to succinctly account for the number of concurrently running components, (ii) Timed automata [2] allow to succinctly model the behavior of programs involving program variables or clocks, and finally (iii) hierarchical systems (also known as hierarchical state machines [3] or hierarchical structures [16]) allow to succinctly represent systems that are decomposed from numerous sub-systems. See also [18] for a recent work, where *web services* are modeled as the asynchronous product of flat systems.

* This work has been partly supported by project ImpRo (ANR-10-BLAN-0317)

An important algorithmic question in this context is whether two given (succinctly presented) systems behave equivalently, or whether one system can be simulated by another one. For instance, if it turns out that a complex system (implementation) is behaviorally equivalent to a simple system (implementation), the system designer can well replace the complex one by the simple one.

Numerous notions of behavioral equivalences have been proposed by van Glabbeek [21, 22]. Among them, *bisimulation equivalence* is undoubtedly the central one of them in formal verification. For instance beautiful characterizations of the bisimulation-invariant fragments of established logics such as first-order logic and monadic second-order have been proven in terms of modal logic [20] and the modal μ-calculus [9], respectively; we refer to [17] for a further such characterization in terms of CTL^*.

Our contributions and related work. In the first part of this paper we study the succinctness with respect to bisimulation equivalence of three established models of succinctly representing finite systems, namely *products of flat systems*, *timed automata*, and *hierarchical systems*. The main contribution of this paper is to pinpoint to the sources of succinctness when comparing any two of the (orthogonally defined) three models of systems, mainly focusing on the different proof ideas for establishing exponential succinctness. We show that each of the three models *can* be exponentially more succinct than any of the other two. Such a rigorous comparison of fundamental models for succinctly representing finite systems has not yet been carried out in the context of formal verification to the best of the authors' knowledge.

In the second part of this paper we study the computational complexity of simulation preorder and bisimulation equivalence checking for products of flat systems, timed systems and hierarchical systems. We provide a general reduction that shows EXPTIME-hardness (and thus completeness) of checking simulation preorder on hierarchical systems and on timed automata. The former is a new result; the latter was already proven in [14] but we believe our proof is more direct. Moreover, our reduction is quite generic, and can be easily applied to a wide range of succinctly presented models.

We also study the problem of deciding simulation preorder and bisimulation equivalence between one of the three above-mentioned succinct systems and a flat system. We show that checking the simulation preorder of a hierarchical system by a flat system is PSPACE-complete. The problem is known to be EXPTIME-complete for (synchronization-free) non-flat systems and timed automata [5].

Via a standard reduction to model checking EF logic we describe a PSPACE algorithm to check bisimilarity of any of the discussed succinctly presented finite systems and a flat system. Essentially since reachability of all of these systems is in PSPACE, it follows that the problem is PSPACE-complete; the upper bound was left open in [5], where it was shown to be PSPACE-hard.

Finally, we study language inclusion between the three above-mentioned succinct models. We show that checking untimed language equivalence (and in fact language universality) is EXPSPACE-hard (and thus EXPSPACE-complete) for hierarchical systems and timed automata. We would like to mention that this

problem has been wrongly cited in the literature as being in PSPACE for timed automata [1, 19]. Our results are summarized in Table 1. Some proofs are omitted due to space constraints; they can be found in [6].

Table 1. Complexity results

	Hierarchical	Timed	Prod. of Flat
Simulation	EXPTIME-c	EXPTIME-c [14]	EXPTIME-c [8]
Bisimulation	PSPACE-hard	EXPTIME-c [14]	EXPTIME-c [10]
Bis. with Flat	PSPACE-c	PSPACE-c	PSPACE-c [5]
Sim. by Flat	PSPACE-c	EXPTIME-c [5]	EXPTIME-c [8, 5]
Language Inc.	EXPSPACE-c	EXPSPACE-c	EXPSPACE-c [14]

2 Definitions

A *transition system* (also *flat system* or just *system*) over a finite alphabet Σ is a tuple $\mathcal{T} = (S, (\xrightarrow{\sigma})_{\sigma\in\Sigma})$, where S is an arbitrary set of *states* and each relation $\xrightarrow{\sigma} \subseteq S \times S$, is the set of σ-labeled *transitions*. Its size is $|\mathcal{T}| = |S| + \sum_\sigma |\xrightarrow{\sigma}|$. We say that an *action* $\sigma \in \Sigma$ is *enabled* at state $s \in S$ if there is a transition $s \xrightarrow{\sigma} s'$ for some $s' \in S$. $\mathcal{T}$ is *deterministic* if each $\xrightarrow{\sigma}$ is a partial function. An *initialized transition system* is $(S, s_0, (\xrightarrow{\sigma})_{\sigma\in\Sigma})$, where $s_0 \in S$ is the *initial state*. A *simulation* is a relation $R \subseteq S \times S$, with the following property: for any states $s, t \in S$ with sRt, for any $\sigma \in \Sigma$ and $s' \in S$ such that $s \xrightarrow{\sigma} s'$, there exists $t' \in S$ with $t \xrightarrow{\sigma} t'$ and $s'Rt'$. A simulation is a *bisimulation* whenever it is symmetric. For two states $s, t \in S$, we write $s \sqsubseteq t$ (resp. $s \sim t$) if there exists a simulation (resp. bisimulation) $R \subseteq S \times S$ such that sRt. An initialized transition system $\mathcal{T} = (S, s_0, (\xrightarrow{\sigma})_{\sigma\in\Sigma})$ is simulated by an initialized transition system $\mathcal{T}' = (S', s'_0, (\xrightarrow{\sigma}{}')_{\sigma\in\Sigma})$, if there is a simulation R in the disjoint union of $\mathcal{T}$ and $\mathcal{T}'$ such that $s_0 R s'_0$. We extend notations $\sqsubseteq$ and $\sim$ to initialized transition systems. We also define $\sim_k$, *bisimilarity up-to k steps*: we have $s \sim_k t$ for two states $s, t \in S$ if, and only if the unfolding of $\mathcal{T}$ from s up-to k steps is bisimilar to the unfolding at t up-to k steps. A path of $\mathcal{T}$ is a sequence of states that are connected by transitions. The length of a path of a transition system is the number of transitions it contains. For a path π, π_i denotes the i-th state it visits, and we denote by $\pi_{i\ldots j}$ the subpath of π from π_i to π_j. For any initialized transition system $\mathcal{T}$, we define $L(\mathcal{T})$ as the *language* accepted by $\mathcal{T}$, that is the set of words made of the transition labels in all paths of $\mathcal{T}$ starting at s_0.

A *product of flat systems* is a tuple $\mathcal{S} = (\mathcal{T}_1, \ldots, \mathcal{T}_k)$, where $\mathcal{T}_i = (S_i, (\xrightarrow{\sigma}_i)_{\sigma\in\Sigma})$ is a flat system for each $1 \leq i \leq k$. $\mathcal{S}$ defines a transition system $\mathcal{T}(\mathcal{S}) = (\prod_i S_i, (\xrightarrow{\sigma})_{\sigma\in\Sigma})$, where $(s_i)_i \xrightarrow{\sigma} (t_i)_i$ if, and only if, for all $1 \leq i \leq k$, either $s_i \xrightarrow{\sigma} t_i$ in $\mathcal{T}_i$, or $t_i = s_i$ and σ is not enabled at s_i. An example is given in Fig. 1.

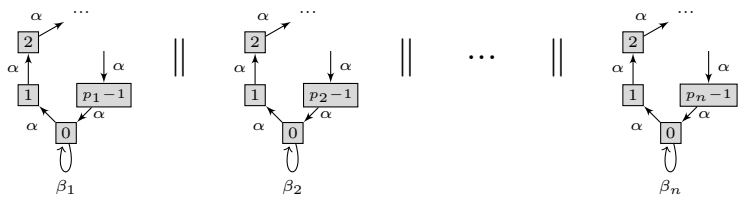

Fig. 1. The system A_n, where $p_1, \ldots, p_n$ are the first n prime numbers, is defined as the product of components F_i made of a α-cycle of length p_i, along where each state corresponds to a value modulo p_i. The self-loop β_i is only available at state 0, which is also the initial state. Then, when the system reads a word α^m, one can read the values $m \mod p_i$ for all $1 \leq i \leq n$, looking at the states of all components.

Hierarchical systems are a modeling formalism used to succinctly describe finite systems, by allowing the reuse of subsystems. A hierarchical system is defined by a simple grammar that generates a single transition system, in which each nonterminal defines a system by explicitly introducing states and transitions, and using other nonterminals. The reuse relation is required to be acyclic, so as to ensure that the generated transition system is finite. These were introduced in [15] in the context of VLSI design.

An *n-pointed system* is a transition system with n selected states, numbered from 1 to n. It is denoted by a pair $(\mathcal{T}, \tau)$, where $\mathcal{T} = (S, (\xrightarrow{\sigma})_{\sigma \in \Sigma})$ is a transition system and $\tau : \{1, \ldots, n\} \to S$ an injection.

Definition 1. *A* hierarchical system *[16] is a tuple $H = (N, I, P)$ where*

1. *N is a finite set of* nonterminals*. Each $B \in N$ has a* rank *denoted by $\mathsf{rank}(B) \in \mathbb{N}$. I is the initial nonterminal with $\mathsf{rank}(I) = 0$.*
2. *P is the set of* productions*, that contains for each $B \in N$ a unique production $B \to (\mathcal{A}, \tau, E)$ where $(\mathcal{A}, \tau)$ is a $\mathsf{rank}(B)$-pointed system with the set of states A, and E is the set of* references *with $E \subseteq \{(B', \sigma) \mid B' \in N, \sigma : \{1, \ldots, \mathsf{rank}(B')\} \to A$ is injective$\}$.*
3. *Define relation $\mathcal{E}_H \subseteq N \times N$ as follows: $(B, C) \in \mathcal{E}_H$ if, and only if for the unique production $B \to (\mathcal{A}, \tau, E)$, E contains some reference of the form (C, σ). We require that $\mathcal{E}_H$ is acyclic.*

Its size is defined as $|H| = \sum_{(B \to (\mathcal{A}, \tau, E)) \in P} |\mathcal{A}| + |E|$. For any production $B \to (\mathcal{A}, \tau, E)$, the states $\tau(i)$ are called *contact states*. Each production produces an n-pointed system, that is, a finite system with n contact states. In fact, a hierarchical system $H = (N, I, P)$ describes a single finite system, obtained by taking, for each production $B \to (\mathcal{A}, \tau, E)$, the disjoint union of the (explicitly given) system $\mathcal{A}$ and those systems defined by nonterminals B' for all references $(B', \sigma) \in E$, and merging the i-th contact state of B' with $\sigma(i)$. Thus, the function σ is used to merge the contact states of the references with the states at the current level. Figure 2 gives an example of a hierarchical system.

Formally, each nonterminal B, produces a $\mathsf{rank}(B)$-pointed system denoted $\mathsf{eval}_H(B)$ (also written as $\mathsf{eval}(B)$ in the rest) as follows. If the production $B \to$

$(\mathcal{A}, \tau, E)$ satisfies $E = \varnothing$, then $\mathsf{eval}(B)$ is the $\mathsf{rank}(B)$-pointed system $(\mathcal{A}, \tau)$. Otherwise, let $E = \{(B_1, \sigma_1), \ldots, (B_k, \sigma_k)\}$ and consider systems $\mathsf{eval}(B_i) = (\mathcal{A}_i, \tau_i)$ for each i. Let $\mathcal{U}$ denote the disjoint union of all $\mathcal{A}_i$ and $\mathcal{A}$. We let $\mathsf{eval}(B) = (\mathcal{U}/_{\equiv}, \pi_{\equiv} \circ \tau)$, where $\equiv$ is the equivalence relation generated by $\{(\sigma_i(j), \tau_i(j)), 1 \leq i \leq k, 1 \leq j \leq \mathsf{rank}(B_i)\}$, and $\pi_{\equiv}$ is the projection to the equivalence classes. Thus, $\equiv$ merges contact state j of system $\mathcal{A}_i$ with the state $\sigma_i(j)$, for each $1 \leq i \leq k$. Note that $\mathsf{eval}(B)$ is well-defined since $\mathcal{E}_H$ is acyclic. We define the generated transition system $\mathcal{T}(H)$ of H as $\mathsf{eval}(I)$.

We denote by $\mathsf{unfolding}(H)$ the tree defined as follows. States are labeled by nonterminals, and the root is the initial nonterminal S. The children of each state labeled by nonterminal B are given as follows. If $B \to (\mathcal{A}, \tau, E)$ is the production of nonterminal B, and if $(B_1, \sigma_1), \ldots, (B_k, \sigma_k)$ are the references in E, then B has a child for each $1 \leq i \leq k$, labeled by B_i. Observe that for each state v of $\mathsf{eval}(H)$, there is a unique state in $\mathsf{unfolding}(H)$ labeled by a nonterminal $B \to (\mathcal{A}, \tau, E)$, such that v is an internal state in $\mathcal{A}$, *i.e.* $v \in \mathcal{A} \setminus \mathsf{range}(\tau)$. We denote this state by $\mathsf{unfolding}(H, v)$

For any nonterminal B in H, an *inner path in* B is a path of $\mathsf{eval}(B)$ that does not contain any contact states of $\mathsf{eval}(B)$, except possibly for the first and the last states. An inner path of B is *traversing* if its first and last states are contact states of $\mathsf{eval}(B)$.

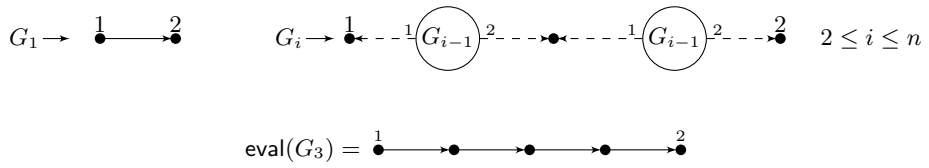

Fig. 2. The figure shows a hierarchical system with nonterminals G_i for $1 \leq i \leq n$. G_1 produces an explicit system with no references, with two contact states (shown by 1 and 2). G_i creates three states, where the leftmost and the rightmost are two contact states, and uses two references to G_{i-1}. The dashed arrows show how to merge the contact states of each copy G_{i-1} with the states of G_i. For instance, the contact state 1 of the leftmost copy of G_{i-1} is merged with contact state 1 of G_i. Then, for $n = 3$, $\mathsf{eval}(G_3)$ is the system depicted on the bottom.

Timed automata are finite automata equipped with a finite set of real-valued clocks. Clocks grow at a constant rate, and are used to enable/disable the transitions of the underlying finite automaton. They can be reset during transitions.

To formally define timed automata, we need the following notations. Given a finite set of clocks $\mathcal{X}$, we call *valuations* the elements of $\mathbb{R}^{\mathcal{X}}_{\geq 0}$. For a subset $R \subseteq \mathcal{X}$ and a valuation v, $v[R \leftarrow 0]$ is the valuation defined by $v[R \leftarrow 0](x) = v(x)$ for $x \in \mathcal{X} \setminus R$ and $v[R \leftarrow 0](x) = 0$ for $x \in R$. Given $d \in \mathbb{R}_{\geq 0}$ and a valuation v, the valuation $v + d$ is defined by $(v + d)(x) = v(x) + d$ for all $x \in \mathcal{X}$. We extend these operations to sets of valuations in the obvious way. We write $\mathbf{0}$ for the valuation that assigns 0 to every clock. An atomic clock constraint is a formula of the form $k \preceq x \preceq' l$ or $k \preceq x - y \preceq' l$ where $x, y \in \mathcal{X}$, $k, l \in \mathbb{Z} \cup \{-\infty, \infty\}$ and

$\preceq, \preceq' \in \{<, \leq\}$. *Guards* are conjunctions of atomic clock constraints. The set $\Phi_{\mathcal{X}}$ denotes the guards over clocks $\mathcal{X}$. A valuation v satisfies a guard g, denoted $v \models g$, if all constraints are satisfied when each $x \in \mathcal{X}$ is replaced with $v(x)$.

Definition 2 ([2]). *A* timed automaton $\mathcal{A}$ *is a tuple* $(\mathcal{L}, \Sigma, \mathcal{X}, \ell_0, E)$*, consisting of finite sets* $\mathcal{L}$ *of locations, a finite alphabet* Σ*,* $\mathcal{X}$ *of clocks,* $E \subseteq \mathcal{L} \times \Phi_{\mathcal{X}} \times \Sigma \times 2^{\mathcal{X}} \times \mathcal{L}$ *of edges, and where* $\ell_0 \in \mathcal{L}$ *is the initial location.*

We are interested in the *time-abstract* semantics of timed automata in the following sense. A timed automaton $\mathcal{A} = (\mathcal{L}, \Sigma, \mathcal{X}, \ell_0, E)$, defines a transition system on the state space $\mathcal{L} \times \mathbb{R}^{\mathcal{X}}_{\geq 0}$, with the initial state $(\ell_0, \mathbf{0})$. There is a transition $(\ell, v) \xrightarrow{\sigma} (\ell', v')$ if, and only if there is $d \geq 0$ and an edge $(\ell, g, \sigma, R, \ell')$ such that $v + d \models g$ and $(v + d)[R \leftarrow 0] = v'$. Although timed automata define infinite transition systems, it is well-known that any timed automaton $\mathcal{A}$ is bisimilar to a computable flat system $\mathcal{T}(\mathcal{A})$, whose size can be exponentially larger than that of $\mathcal{A}$ [2]. See Figure 3 for a timed automaton bisimilar to a large flat system.

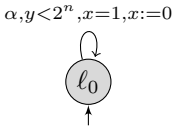

Fig. 3. A timed automaton with one location and two clocks x, y, modelling a "counter" ranging from 0 to 2^n that can only be incremented. At any state (ℓ_0, v) with $v(x) = 0$, $v(y)$ encodes the value of the counter. Taking the self-loop increments the counter. Any run stops after at most 2^n increments.

3 Succinctness

We compare hierarchical systems, products of flat systems, and timed automata with respect to succinctness of models. Our results show that these classes are pairwise incomparable in terms of succinctness: inside each class, there are infinite families of models which are exponentially more succinct than any bisimilar family of models in another class.

3.1 Hierarchical Systems *vs.* Products of Flat Systems

We show that hierarchical systems can be exponentially more succinct than products of flat systems: it is easy to define long finite chains with the former, as in Fig. 2, although this is not possible with the latter.

Theorem 1. *Hierarchical systems can be exponentially more succinct than products of flat systems.*

The other direction, in the next theorem, is more difficult.

Theorem 2. *Products of flat systems can be exponentially more succinct than hierarchical systems.*

This theorem also establishes a non-trivial property of hierarchical systems, giving insight into the differences with the other classes. It shows that any hierarchical graph of size n defining an exponentially large graph contains necessarily two states that are bisimilar up-to $\Omega(n)$ steps. The proof is based on the observation that this is not the case for products of flat systems.

We consider the system A_n described in Fig. 1. We first give simple properties of A_n, based on the Chinese Remainder Theorem:

Theorem 3 (Chinese Remainder Theorem). *Let $p_1, \dots, p_n$ denote pairwise coprime numbers. For any integers $a_1, \dots, a_n$, there exists a unique $m \in [0, p_1 p_2 \dots p_n - 1]$ such that $m \equiv a_i \mod p_i$ for all $1 \leq i \leq n$.*

Let $p_1, \dots, p_n$ denote the first n prime numbers, and consider A_n given as the product of components $F_1, \dots, F_n$. Observe that A_n has $p_1 p_2 \cdots p_n$ states. By the Chinese Remainder Theorem, all bisimulation classes of A_n are singletons. In fact, consider a state s reached from the initial state by reading α^m. While executing the word α^{p_n} from s, measuring the minimal distance to some state that enables β_i, we can deduce m modulo p_i for each $1 \leq i \leq n$, and this uniquely determines m modulo $p_1 \cdots p_n$. This is formalized in the following lemma. In the rest of the section, we refer to states of A_n by natural numbers from 0 to $p_1 p_2 \cdots p_n - 1$.

Lemma 1. *For any pair of states $0 \leq c < c' < p_1 p_2 \cdots p_n$ of A_n, $c \not\sim_{p_n} c'$. Moreover, $c \sim_{p_n} c'$ if, and only if $c \sim c'$.*

Let us first explain the idea of the proof of Theorem 2. Any (sufficiently large) hierarchical graph of size polynomial in n that is bisimilar to A_n contains two occurrences of a nonterminal since A_n contains an exponential number of states. Similarly, some contact states of a same nonterminal appear several times. We will show moreover that for any hierarchical graph, there is a bisimilar one whose size is polynomially bounded, with the property that for some nonterminal B, the same contact state of two different copies of $\mathsf{eval}(B)$ belong to inner paths of $\mathsf{eval}(B)$ that are bisimilar up-to p_n steps. Thus, both occurrences of the contact state must be bisimilar to the same state of A_n by Lemma 1. However, we also show that these are reachable from the initial states in less than $p_1 p_2 \cdots p_n$ steps, which leads to a contradiction. We now give a formal proof following these ideas.

The size of A_n can be seen to be polynomially bounded in n from below and above since $p_n \sim n \log(n)$ for large n by the Prime Number Theorem. Assume there are hierarchical systems H_n such that $\mathcal{T}(H_n) \sim \mathcal{T}(A_n)$ for all $n \geq 0$, such that $|H_n| \leq f(n)$ for some polynomial f. We first show, in the following lemma, that each H_n can be assumed to satisfy the following Property ($\star$): for all nonterminals B, all traversing paths of $\mathsf{eval}(B)$ have length either 0 or at least $p_n + 2$.

Lemma 2. *For any family $(H_n)_{n \geq 0}$ of pointed hierarchical systems, there exist pointed hierarchical systems $(H'_n)_{n \geq 0}$ such that $|H'_n| \leq p(|H_n|)$ for some polynomial p, $\mathcal{T}(H_n) \sim \mathcal{T}(H'_n)$ and H'_n satisfies Property ($\star$), for all $n \geq 0$.*

The idea of the transformation is the following. We consider each production $B \to (\mathcal{A}, \tau, E)$, in the reverse topological order w.r.t. $\mathcal{E}_{H_n}$. Then, for all productions $C \to (\mathcal{A}', \tau', E')$ with $(C, B) \in \mathcal{E}_{H_n}$, we add to $\mathcal{A}'$ a copy of each traversing path ρ of $\mathcal{A}$ of size less than $p_n + 2$, and remove all edges labeled by α leaving the first state of ρ. The construction is illustrated in Fig. 4.

Proof of Theorem 2. We assume that Property $(\star)$ holds, by Lemma 2. Consider any $n \geq 0$, such that $12f(n)^4 < p_1 p_2 \cdots p_n$, and let us write $H_n = (N, I, P)$. Consider a path π obtained in H_n by reading $\alpha^{p_1 \cdots p_n}$ from the initial state. For all nonterminals $B \in N$, with the unique production $B \to (\mathcal{A}, \tau, E)$, and $i \in \{1, \ldots, \mathsf{range}(\tau)\}$, let us mark by (B, i) in π all states that are equivalent, under relation $\equiv$, to the state $\tau(i)$ of $\mathcal{A}$. A single state in $\mathsf{eval}(H_n)$ can be marked by several pairs (B, i) since the equivalence $\equiv$ merges states. For example, if we were to apply this marking to the graph of Fig. 2, then the leftmost state would be marked by $(G_1, 1), (G_2, 1), \ldots, (G_n, 1)$, since at each production G_i, this state is merged with contact state 1 of the leftmost occurrence of nonterminal G_{i-1}. Note that at least one state among any consecutive $|H_n|$ states must be marked by some (B, i) in π. Otherwise π would not visit any contact states, and therefore would stay inside the same explicit graph, which has size less than $|H_n|$, and some state would appear twice since $p_1 p_2 \cdots p_n > f(n) \geq |H_n|$. Then, a same state of $\mathcal{T}(H_n)$ would be bisimilar to two distinct states of $\mathcal{T}(A_n)$, which contradicts Lemma 1. Since the number of pairs (B, i) is bounded by $|H_n|$ at least $m = \frac{|\pi|}{|H_n|^2}$ states of π are marked by some pair (B, i). Observe that $m = |\pi|/|H_n|^2 = \Omega(2^n/f(n)^2)$. Now, at least half the states marked by (B, i) mark the beginning of traversing paths of $\mathsf{eval}(B)$.

Among these states, assume that there are π_j and $\pi_{j'}$ for $0 \leq j < j' < p_1 \cdots p_n$, such that the traversing paths starting at these states have positive length (therefore, at least $p_n + 2$, thus contain at least p_n inner states). By assumption, these states are bisimilar to the states of A_n corresponding to the numbers j and j' respectively. Consider the inner paths $\pi_{j \ldots j+p_n}$ and $\pi_{j' \ldots j'+p_n}$ of $\mathsf{eval}(B)$. These paths belong to different instances of the production of B, so the visited states are pairwise disjoint. However, states π_j and $\pi_{j'}$, seen as states of $\mathsf{eval}(B)$ are bisimilar, since they correspond to the same contact state of $\mathsf{eval}(B)$. Since all α-labelled transitions from π_j lead to bisimilar states in $\mathsf{eval}(B)$, all internal paths starting at π_j are bisimilar. In particular, π_j and $\pi_{j'}$ are bisimilar up-to p_n steps, since they stay inside $\mathsf{eval}(B)$. So, each β_i is enabled in π_{j+k} iff it is enabled at $\pi_{j'+k}$, for all $1 \leq k \leq p_n$. But then π_j and $\pi_{j'}$ must be bisimilar to the same state of A_n by Lemma 1, and this is a contradiction.

Assume now that there is no more than one state π_j marked by (B, i) with a positive-length traversing path; so there are at least $m/2 - 1$ states corresponding to beginnings of traversing paths of length 0 (consisting of single states). Let $(\alpha_j)_{1 \leq j}$ denote the indices such that π_{α_j} is marked by (B, i) and is a traversing path of length 0. We argue that some state marked by a pair (B', i') with $(B, i) \neq (B', i')$, that is the beginning or the end of a traversing path of positive length must occur in $\pi_{\alpha_1 \ldots \alpha_1 + |H_n|}$. Consider the nonterminal C labeling the state $\mathsf{unfolding}(H_n, \pi_{\alpha_1})$. By definition, π_{α_1} is an inner state of C, so π_{α_1} is part of

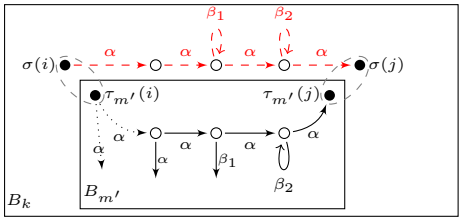

Fig. 4. The construction of Theorem 2 that removes a small traversing path created in a production $B_{m'} \to (\mathcal{A}_{m'}, \tau_{m'}, E_{m'})$. Here, $B_{m'}$ has an internal path of length 4 between $\tau_{m'}(i)$ and $\tau_{m'}(j)$, where internal states are represented by unfilled states. The contact states $\tau_{m'}(i)$ and $\tau_{m'}(j)$ are to be merged with the states $\sigma(i)$ and $\sigma(j)$ created in the production of B_k. The construction removes all edges of $\mathcal{A}_{m'}$ leaving $\tau_{m'}(i)$ (shown by dotted arrows). Then, the red dashed path ρ' is added instead from $\sigma(i)$ and $\sigma(j)$ in $\mathcal{A}_k$.

an inner path of $\mathsf{eval}(C)$. Since the structure defined in the production of C has size less than $|H_n|$, $\pi_{\alpha_1 \ldots \alpha_1 + |H_n|}$ must visit a state labeled by (B', i') that is the beginning or the end of an inner path of positive length: this is either a contact state of $\mathsf{eval}(C)$ (the end of the inner path containing π_{α_1}), or the beginning of an inner path inside $\mathsf{eval}(B')$, where $(C, B') \in \mathcal{E}_{H_n}$. In fact, if this subpath does not visit contact states of C and if it only contains inner paths of length 0 for other nonterminals B', then it stays inside the explicit graph defined in the production of C. This is again a contradiction with the bisimilarity with A_n since a state then must appear twice. This shows that every chunk of $|H_n|$ starting at some π_{α_i} contains a state marked by some other (B', i'), which is the beginning or the end of some traversing path of positive length of $\mathsf{eval}(B')$. Then, at least $\frac{m/2-1}{2|H_n|^2}$ states are marked by the same (B', i'), and are the beginning of traversing paths of positive length, and we can apply the previous case.

3.2 Timed Automata *vs.* Product of Flat Systems

Theorem 4. *Timed automata can be exponentially more succinct than products of flat systems.*

Proof. The proof immediately follows from Theorem 1 and the fact that the timed automaton of Fig. 3 is time-abstract bisimilar to the system G_n. □

We now show that products of flat systems can be more succinct than timed automata. This result requires new techniques since the nature of state-space explosion of timed automata is different; it is due to the complex relation between its clock values, rather than to its structure. To show this result, we use the well-known notion of *zones*, which are convex sets of the state space with integer corners. We only need the fact that zones are closed under basic operations such as time predecessors and intersection. We refer to [4] for definitions and properties of zones.

The main idea behind the proof is the following: a state in the transition system defined by a timed automaton can have an exponential number of a priori pairwise non-bisimilar successors but we show that the pairwise non-bisimilarity of an exponential number of successors of a state cannot be detected by looking only one step further in the transition system. This important property is established using geometric properties of regions, and it is inherent to transition systems defined by timed automata. We show, on the other hand, that such a system can be defined by a small product of automata (system A'_n defined below), which yields the following theorem.

Theorem 5. *Products of flat systems can be exponentially more succinct than timed automata.*

For any $n \geq 1$, we define the finite transition system $\mathcal{T}_n$ on the set of states $S_n = \{(c_1, \ldots, c_n) \mid \forall 1 \leq i \leq n, 0 \leq c_i < p_i\}$, where p_i is the $i+2$-th prime number (so that we have $p_i \geq 5$, see below). From any state $(c_1, \ldots, c_n) \in S_n$ and any vector $(b_1, \ldots, b_n) \in \{1, 2\}^n$, there is a transition $(c_1, \ldots, c_n) \xrightarrow{\alpha} (c_1 + b_1 \mod p_1, \ldots, c_n + b_n \mod p_n)$. Moreover, we have a self-loop $(c_1, \ldots, c_n) \xrightarrow{\beta_i} (c_1, \ldots, c_n)$ whenever $c_i \equiv 0 \mod p_i$. $\mathcal{T}_n$ can be defined by adapting the system A_n of Fig. 1, by adding an edge from x to $x + 2$ (modulo p_i) inside each component F_i. Let us call A'_n this product of flat systems. It is clear that A'_n has size $O(n^2 \log(n))$ since $p_n \sim n \log n$.

The following lemma shows that states of $\mathcal{T}_n$ cannot simulate each other.

Lemma 3. *For all states $\boldsymbol{c}, \boldsymbol{c'}$ of $\mathcal{T}_n$, there is no simulation R such that $\boldsymbol{c}\ R\ \boldsymbol{c'}$.*

Proof (of Thm. 5). We consider any timed automaton T_n bisimilar to $\mathcal{T}_n$ (thus, to A'_n). By definition, all states of $\mathcal{T}_n$ have 2^n transitions, all leading to pairwise non-bisimilar states. We show that such a branching is not possible in T_n unless T_n has exponential size.

We consider the state $\boldsymbol{c} = (p_1 - 1, \ldots, p_n - 1)$ of $\mathcal{T}_n$, which is reachable. Let (ℓ, v) be any state of T_n that is bisimilar to $\boldsymbol{c}$. In $\mathcal{T}_n$, $\boldsymbol{c}$ has 2^n α-successors. Moreover, for each $1 \leq i \leq n$, β_i is enabled in exactly half of these successor states. In fact, for any subset $P \subseteq \{\beta_1, \ldots, \beta_n\}$, there is a successor where the set of enabled transition labels is exactly $P \cup \{\alpha\}$. Let $E(\ell)$ denote the number of edges from ℓ. For each successor $\boldsymbol{c'}$ of $\boldsymbol{c}$, pick a transition from (ℓ, v) in T_n, leading to a state bisimilar to $\boldsymbol{c'}$. Then, at least $2^n/E(\ell)$ of these transitions are along some edge $e = (\ell, \phi, \alpha, R, \ell')$. This means that there exist $d_1, \ldots, d_m \geq 0$ with $m = \lfloor 2^n/E(\ell) \rfloor$ such that states $(\ell, v+d_i)$ satisfy the guard ϕ; and the states $(\ell', v'_i) = (\ell', (v + d_i)[R \leftarrow 0])$ are each bisimilar to a successor of $\boldsymbol{c}$. States (ℓ, v'_i) are therefore pairwise non-simulating, by Lemma 3. Let us note here that R cannot be empty, since otherwise $(\ell', v + d_i)$ can simulate $(\ell', v + d_j)$ whenever $d_i \leq d_j$, which contradicts Lemma 3. We are going to show that there must be $\Omega(2^n)$ edges leaving ℓ'.

Valuations $v + d_i$ belong to a line of direction $\mathbf{1}$, that contains v. So the projections $v'_i = (v + d_i)[R \leftarrow 0]$ also belong to a line $\mathcal{D}$. Consider the set $g_1, \ldots, g_m$ of guards of the edges leaving ℓ'. Such a transition can be taken from

v'_i if, and only if $v'_i + d \in g_j$ for some delay $d \geq 0$. This condition is equivalent to $v'_i \in \bigwedge_{x \in R}(x = 0) \wedge \mathsf{Pre}(g_i)$, where Pre gives the set of *time-predecessors* of g_i, *i.e.* $\mathsf{Pre}(g_i) = \{v \mid \exists d \geq 0, v + d \models g_i\}$. It is well-known that the right hand side of the above expression can be expressed by a guard [4]. Therefore, for simplicity, but without loss of generality, let us replace g_i by the right hand side of the above. Thus, we have now a line $\mathcal{D}$ that contains all valuations v'_i, and convex sets defined by the guards. The intersection of each guard with $\mathcal{D}$ is a segment. From now on, we are only interested in valuations and segments that lie in $\mathcal{D}$. Each segment along $\mathcal{D}$ thus can be seen as an interval.

Now, we will show that only a small number of bisimulation classes can be distinguished inside $\mathcal{D}$, looking only at the immediate enablement of m guards. For a set of real intervals $\mathcal{I} = \{I_1, \ldots, I_n\}$, we denote by $\chi_{\mathcal{I}}$ the equivalence relation among real numbers given by $(x, y) \in \chi_{\mathcal{I}}$ if, and only if $x \in I \Leftrightarrow y \in I$, for all $I \in \mathcal{I}$. When $\mathcal{I}$ is finite, this relation is finite too. For instance, if $\mathcal{I} = \{[a, b]\}$, then $\chi_{\mathcal{I}}$ has index 2. We denote by $|\chi_{\mathcal{I}}|$ the index of $\chi_{\mathcal{I}}$.

Lemma 4. *Let $\mathcal{I}$ be a finite set of real intervals, and let J be a real interval. Then $|\chi_{\mathcal{I} \cup \{J\}}| \leq |\chi_{\mathcal{I}}| + 2$.*

Now, using m guards, one can only define $2m$ subsets of $\mathcal{D}$ which are pairwise distinguished with respect to the satisfaction of all guards g_i. By the previous lemma, there are at most $2m$ equivalence classes defined by $\chi_{\{g_1, \ldots, g_{E(\ell')}\}}$. On the other hand, any pair of states v'_i and v'_j can be distinguished by the satisfaction of some guard g_k, since this is the case for the 2^n successors of $\boldsymbol{c} = (p_1 - 1, \ldots, p_n - 1)$ inside $\mathcal{T}_n$. It follows that $2m \geq 2^n / E(\ell)$. Therefore, $|T_n| \geq m = \Omega(2^n)$. □

3.3 Timed Automata *vs.* Hierarchical Systems

Theorem 6. *Timed automata can be exponentially more succinct than hierarchical systems, and vice versa.*

The proof of the first direction is similar to that of Theorem 2: we give a timed automaton that describes a system similar to A_n. The other direction uses the techniques of Theorem 5.

4 Complexity of Preorder Checking

4.1 Hardness of Simulation

The main result of this section is that deciding simulation between two hierarchical systems is EXPTIME-complete. Our proof is based on a simple reduction from countdown games [11]. Our reduction is quite generic, and it can be applied to any class with a set of simple properties (discussed at the end of the section). As an example, we apply the reduction to timed automata. Note the EXPTIME-hardness of checking simulation for timed automata was already proved in [14] by a reduction based on Turing machines; we obtain here a simpler proof.

Theorem 7. *Checking simulation between two hierarchical systems (resp. two timed automata) is* EXPTIME*-complete.*

Our reduction is based on *countdown games* [11], defined as follows. A countdown game $\mathcal{C}$ is played on a weighted graph (S, T), whose edges are labeled with positive integer weights encoded in binary. A *move* of the game from configuration $(s, c) \in S \times \mathbb{N}$ is determined jointly by both players, as follows. First, Eve chooses a number $d \leq c$ such that $(s, d, s') \in T$ for some state s'. Then Adam chooses a state $s' \in S$ such that $(s, d, s') \in T$. The resulting configuration is $(s', c-d)$. The game stops when Eve has no available moves: configuration (s, c) is *winning* for Eve if $c = 0$. Given a countdown game, one can build an equivalent turn-based graph game of exponential size with a reachability objective. We note that given a countdown game and an initial configuration, the existence of a winning strategy for Eve is EXPTIME -complete [11].

We first reduce the problem of determining the winner in countdown games to the simulation problem on finite automata, that may have exponential size. We then show how these automata can be described in polynomial size by hierarchical systems and timed automata. This proves the EXPTIME-hardness (thus, completeness) of the simulation problem on these classes.

Consider a countdown game $\mathcal{C} = (S, T)$ with initial state $q \in S$ and initial value c. Let Σ denote the set of constants used in $\mathcal{C}$. We define two finite automata on the alphabet $\Gamma = \Sigma \cup \{e, \alpha, \beta\}$. The first one, called $\text{Counter}_{\mathcal{C}}(c)$, is a directed path of length c with some additional states, defined in Fig. 5. The bottom left state is the initial state. Intuitively, this is used to count down from c when simulating the countdown game.

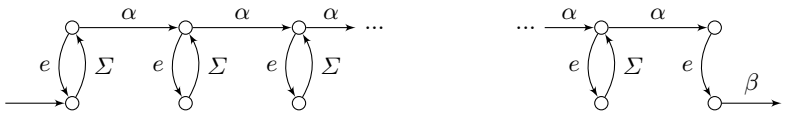

Fig. 5. System $\text{Counter}_{\mathcal{C}}(c)$.

The second automaton is called $\text{Control}_{\mathcal{C}}$, and has the same structure as the game $\mathcal{C} = (S, T)$, except that each transition labeled by $k \in \Sigma$ is replaced by a module $\text{Chain}_{\mathcal{C}}(k)$, which is roughly a directed path of length $k + 1$ labeled by α^k. In addition, in every state, an edge leads to a sink state by any symbol of $\Gamma \setminus \{\alpha\}$ from all but the last state, and by any symbol in $\Gamma \setminus \{e\}$ from the last state. Sink states have self-loops on all symbols. The module is given in Fig. 6.

Now, automaton $\text{Control}_{\mathcal{C}}$ is defined by replacing each transition labeled by k in the game $\mathcal{C}$, with an instance of module $\text{Chain}_{\mathcal{C}}(k)$, as shown in Fig. 7. Moreover, from each state s_i, there is an edge going to a sink state, labeled by all labels in $\Gamma \setminus (\Sigma(s_i) \cup \{\beta\})$, where $\Sigma(s_i)$ denotes the set of labels of the edges leaving s_i in game $\mathcal{C}$. This ensures that a path in $\text{Control}_{\mathcal{C}}$ encodes a correct simulation of the game. The initial state of $\text{Control}_{\mathcal{C}}$ is the initial state of $\mathcal{C}$.

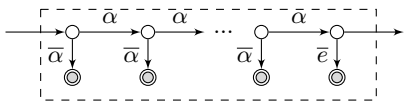

Fig. 6. Module $\text{Chain}_{\mathcal{C}}(k)$. Here, $\overline{x}$ denotes the complement of the set $\{x\}$. All gray states at the bottom in the figure are sink states with (omitted) self-loops on all symbols.

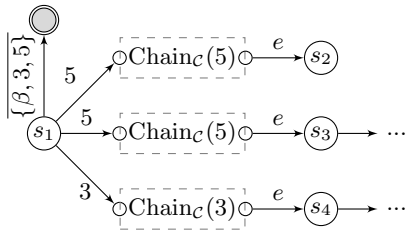

Fig. 7. A part of $\text{Control}_{\mathcal{C}}$ for a countdown game $\mathcal{C}$ with states s_1, s_2, s_3, s_4 and edges $(s_1, 5, s_2), (s_1, 5, s_3), (s_1, 3, s_4)$.

Proposition 1. *For any countdown game $\mathcal{C} = (S, T)$ with initial state s_1 and initial value c, $\text{Counter}_{\mathcal{C}}(c) \sqsubseteq \text{Control}_{\mathcal{C}}$ if, and only if Eve does not have a winning strategy in $\mathcal{C}$ from configuration (s_1, c).*

We now explain how this reduction can be applied to hierarchical systems and timed automata, in polynomial time. For hierarchical systems, in order to succinctly represent $\text{Counter}_{\mathcal{C}}(c)$ and modules $\text{Chain}_{\mathcal{C}}(k)$, we use the trick of Fig. 2. For instance, in order to define $\text{Chain}_{\mathcal{C}}(k)$, one can generate all systems $G_1, G_2, \ldots, G_{\lceil \log(k) \rceil}$ and combine these according to the binary representation of k. For timed automata, a pair of clocks can be simply used to count upto k, as in Fig. 3. Thus, modules $\text{Counter}_{\mathcal{C}}(c)$ and $\text{Chain}_{\mathcal{C}}(k)$ can be defined in polynomial space in these classes, which yields a polynomial-time reduction.

4.2 Simulation and Bisimulation with a Flat System

We show that checking whether a hierarchical graph is simulated by a finite automaton is PSPACE-complete. The PSPACE-membership follows from the fact that simulation by a finite automaton can reduced to μ-calculus model-checking (see e.g. [13]), which is in turn in PSPACE [7]. The corresponding lower bound can in fact be deduced from results from [12] and [13] by using lengthy definitions, however, we decided to give a direct reduction from quantified Boolean satisfiability problem.

Theorem 8. *Checking whether a flat system simulates a hierarchical system is* PSPACE*-complete.*

Second, we show that the problems of checking bisimilarity between a timed automaton and a flat system, and between a hierarchical system and a flat system are PSPACE-complete. In fact, one can reduce bisimilarity with a flat system to model checking CTL's fragment EF (where formulas are represented as DAGs) in polynomial time [13]. This yields a polynomial space algorithm for this problem since EF model-checking is easily seen to be in PSPACE for products of flat systems, timed automata and hierarchical systems since reachability for all these systems is in PSPACE.

Theorem 9. *Checking bisimilarity between a timed automaton (resp. hierarchical system, product of flat systems) and a flat system is* PSPACE*-complete.*

The PSPACE-hardness for product of finite automata was already proved in [5]. For timed automata, it follows from PSPACE-hardness of control state reachability that checking *any* relation between time-abstract language equivalence and time-abstract bisimulation between a timed automaton and a finite automaton is PSPACE-hard. For hierarchical systems, we observe that the reduction of [13] that shows the PSPACE-hardness of checking bisimulation between a pushdown automaton and a finite automaton can be adapted to hierarchical systems.

4.3 Language Inclusion and Universality

Given any timed automaton $\mathcal{A}$, one can effectively construct an exponential-size finite automaton, called the *region automaton* that is time-abstract bisimilar to $\mathcal{A}$ [2]. Then, using region automata, one can decide the inclusion between the untimed languages of two timed automata in exponential space. The exact complexity of these problems had not been characterized, and the problem was wrongly cited in the literature as being PSPACE in [1, 19]. In this section, we prove that untimed language universality and inclusion are actually EXPSPACE-complete. The result holds already for two clocks. For one clock, the problem is PSPACE-complete. Language inclusion can also be decided in exponential space for hierarchical systems, since the system generated has at most exponential size. We adapt the proof to hierarchical systems, and obtain the same complexity results.

Theorem 10. *Checking untimed language universality is* EXPSPACE*-complete for timed automata with two clocks, and* PSPACE*-complete with one clock. Language universality is* EXPSPACE*-complete for hierarchical systems.*

To prove this, we consider the acceptance problem on exponential-space Turing machines, and show how to compute timed automata (resp. hierarchical systems) that accept all words but those encoding correct accepting executions.

5 Conclusion

In this paper, we compared products of automata, timed automata and hierarchical systems, which are used to succinctly describe finite-state systems. We showed that each of them contains models that are exponentially more succinct than the others, formalizing the intuition that the nature of the state space explosion is different in each formalism. Several variants of these systems were not considered in this paper. For instance, silent transitions improve succinctness in general: the main argument in the proof of Theorem 5 does not hold for timed automata with silent transitions. One could also study different synchronization semantics for products of automata. We also studied the computational complexity of several preorder and equivalence relations. The complexity of bisimilarity between hierarchical systems remains open.

References

1. Luca Aceto, Anna Ingólfsdóttir, Kim Guldstrand Larsen, and Jiri Srba. *Reactive Systems: Modelling, Specification and Verification.* Cambridge University Press, NY, USA, 2007.
2. Rajeev Alur and David L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126:183–235, 1994.
3. Rajeev Alur and Mihalis Yannakakis. Model checking of hierarchical state machines. *ACM Trans. Program. Lang. Syst.*, 23(3):273–303, 2001.
4. Johan Bengtsson and Wang Yi. Timed automata: Semantics, algorithms and tools. In *Lectures on Concurrency and Petri Nets*, p. 87–124, 2003.
5. Laura Bozzelli, Axel Legay, and Sophie Pinchinat. Hardness of preorder checking for basic formalisms. In *LPAR (Dakar)*, p. 119–135, 2010.
6. Romain Brenguier, Stefan Göller, and Ocan Sankur. A comparison of succinctly represented finite-state systems. Technical Report LSV-12-13, Lab. Specification & Verification, ENS Cachan, France, June 2012.
7. Stefan Göller and Markus Lohrey. Fixpoint logics over hierarchical structures. *Theory Comput. Syst.*, 48(1):93–131, 2011.
8. David Harel, Orna Kupferman, and Moshe Y. Vardi. On the complexity of verifying concurrent transition systems. *Inf. Comput.*, 173:143–161, March 2002.
9. David Janin and Igor Walukiewicz. On the Expressive Completeness of the Propositional mu-Calculus with Respect to Monadic Second Order Logic. In *Proc. of CONCUR*, LNCS 1119, p. 263–277. Springer, 1996.
10. Lalita Jategaonkar and Albert R. Meyer. Deciding true concurrency equivalences on safe, finite nets. *Theoretical Computer Science*, 154(1):107–143, 1996.
11. Marcin Jurdziński, François Laroussinie, and Jeremy Sproston. Model checking probabilistic timed automata with one or two clocks. In TACAS'07, LNCS 4424, p. 170–184. Springer, March 2007.
12. Antonín Kučera and Richard Mayr. Why is simulation harder than bisimulation? In *CONCUR*, LNCS 2421, p. 594–610. Springer, 2002.
13. Antonín Kučera and Richard Mayr. On the complexity of checking semantic equivalences between pushdown processes and finite-state processes. *Information and Computation*, 208(7):772–796, 2010.
14. François Laroussinie and Philippe Schnoebelen. The state explosion problem from trace to bisimulation equivalence. In FOSSACS'07, p. 192–207. Springer, 2000.
15. Thomas Lengauer and Egon Wanke. Efficient solution of connectivity problems on hierarchically defined graphs. *SIAM J. Comput.*, 17(6):1063–1080, 1988.
16. Markus Lohrey. Model-checking hierarchical structures. *J. Comput. Syst. Sci.*, 78(2):461–490, 2012.
17. Faron Moller and Alexander Moshe Rabinovich. Counting on CTL*: on the expressive power of monadic path logic. *Inf. Comput.*, 184(1):147–159, 2003.
18. Anca Muscholl and Igor Walukiewicz. A lower bound on web services composition. *Logical Methods in Computer Science*, 4(2), 2008.
19. Jiri Srba. Comparing the expressiveness of timed automata and timed extensions of Petri nets. In FORMATS'08, LNCS 5215, p. 15–32. Springer, 2008.
20. Johan van Benthem. *Modal Correspondence Theory.* PhD thesis, University of Amsterdam, 1976.
21. Rob J. van Glabbeek. The linear time-branching time spectrum (extended abstract). In *CONCUR*, LNCS 458, p. 278–297. Springer, 1990.
22. Rob J. van Glabbeek. The linear time - branching time spectrum ii. In *CONCUR*, LNCS 715, p. 66–81. Springer, 1993.