Theoretical Computer Science 637 (2016) 1–29

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

Forward analysis and model checking for trace bounded WSTS ☆

Pierre Chambart [a], Alain Finkel [b,*], Sylvain Schmitz [b]

[a] OCamlPro, France
[b] LSV, ENS Cachan & CNRS & INRIA, Université Paris-Saclay, France

ARTICLE INFO

Article history:
Received 25 June 2015
Received in revised form 4 March 2016
Accepted 15 April 2016
Available online 19 April 2016
Communicated by J.-F. Raskin

Keywords:
Complete WSTS
Model checking
Flattable system
Bounded language
Acceleration

ABSTRACT

We investigate a subclass of well-structured transition systems (WSTS), the *trace bounded*—in the sense of Ginsburg and Spanier (1964), [1]—complete deterministic ones, which we claim provide an adequate basis for the study of forward analyses as developed by Finkel and Goubault-Larrecq (2012), [2]. Indeed, we prove that, unlike other conditions considered previously for the termination of forward analysis, trace boundedness is decidable. Trace boundedness turns out to be a valuable restriction for WSTS verification, as we show that it further allows to decide all ω-regular properties on the set of infinite traces of the system.

1. Introduction

General context. Forward analysis using acceleration [3,4] is established as one of the most efficient practical means—albeit in general without termination guarantee—to tackle safety problems in infinite state systems, e.g. in the tools TReX [5], Lash [6], or Fast [7]. Even in the context of *well-structured transition systems* (WSTS), a unifying framework for infinite systems endowed with a generic backward coverability algorithm due to Abdulla et al. [8], forward procedures are commonly felt to be more efficient than the backward algorithm [9]: e.g. for lossy channel systems [10], although the backward procedure always terminates, only the non-terminating forward procedure is implemented in the tool TReX [5].

Acceleration techniques rely on symbolic representations of sets of states to compute exactly the effect of repeatedly applying a finite sequence of transitions w, i.e. the effect of w^*. The forward analysis terminates if and only if a finite sequence $w_1^* \cdots w_n^*$ of such accelerations deriving the full reachability set can be found, resulting in the definition of the *post* flattable* class of systems [4]. Despite evidence that many classes of systems are flattable [11–13], whether a system is post*-flattable is undecidable for general systems [4].

The well structured case. Finkel and Goubault-Larrecq [14,2] have laid new theoretical foundations for the forward analysis of deterministic WSTS—where determinism is understood with respect to transition labels—by defining *complete* deterministic WSTS (cd-WSTS) as a means to obtain finite representations for downward closed sets of states [see also [15]], ∞-*effective* cd-WSTS as those for which the acceleration of certain sequences can effectively be computed, and by proposing a concep-

☆ Work supported by ANR projects AVeriSS (ANR-06-SETIN-001), AVERILES (ANR-05-RNTL-02) and REACHARD (ANR-11-BS02-001).
* Corresponding author.
E-mail address: Alain.Finkel@lsv.ens-cachan.fr (A. Finkel).

http://dx.doi.org/10.1016/j.tcs.2016.04.020
0304-3975/

tual forward procedure à la Karp and Miller [16] for computing the full cover of a cd-WSTS—i.e. the downward closure of its set of reachable states. Similarly to post* flattable systems, this procedure called "Clover" terminates if and only if the cd-WSTS at hand is *cover flattable*, which is undecidable [2]. As we show in this paper, post* flattability is also undecidable for cd-WSTS, thus motivating the search for even stronger sufficient conditions for termination. A decidable sufficient condition that we can easily discard as too restrictive is trace set finiteness, corresponding to terminating systems [17].

This work. Our aim with this paper was to find a reasonable decidable sufficient condition for the termination of the Clover procedure. We have found one such condition in the work of Demri, Finkel, Goranko, and van Drimmelen [18] with *trace flattable* systems, which are maybe better defined as the systems with a *bounded* trace set in the sense of Ginsburg and Spanier [1]: a language $L \subseteq \Sigma^*$ is bounded if there exist $n \in \mathbb{N}$ and n words $w_1, \ldots, w_n$ in Σ^* such that $L \subseteq w_1^* \cdots w_n^*$. The regular expression $w_1^* \cdots w_n^*$ is called a *bounded expression* for L. Trace bounded cd-WSTS encompass systems with finite trace set.

Trace boundedness implies post* and cover flattability. Moreover, Demri et al. show that it allows to decide liveness properties for a restricted class of counter systems (see also [19,20] for other classes of trace bounded counter systems). However, to the best of our knowledge, nothing was known regarding the decidability of trace boundedness itself, apart from the 1964 proof of decidability for context-free grammars by Ginsburg and Spanier [1] and the 1969 one for equal matrix grammars by Siromoney [21].

We characterize trace boundedness for cd-WSTS and provide as our main contribution a generic decision algorithm in Section 3. We employ vastly different techniques than those used by Ginsburg and Spanier [1] and Siromoney [21], since we rely on the results of Finkel and Goubault-Larrecq [14,2] to represent the effect of certain transfinite sequences of transitions. We further argue in Section 4 that both the class of systems (deterministic WSTS) and the property (trace boundedness) are in some sense optimal: we prove that trace boundedness becomes undecidable if we relax either the determinism or the well-structuredness conditions, and that the less restrictive property of post* flattability is not decidable on deterministic WSTS.

We investigate in Section 5 the complexity of trace boundedness. It can grow very high depending on the type of underlying system, but this is the usual state of things with WSTS—e.g. the non-multiply-recursive lower bound for coverability in lossy channel systems of Chambart and Schnoebelen [22] also applies to trace boundedness—and does not prevent tools to be efficient on case studies. Although there is no hope of finding general upper bounds for all WSTS, we nevertheless propose a generic proof recipe, based on a detailed analysis of our decidability proof, which results in tight upper bounds in the cases of lossy channel systems and affine counter systems. In the simpler case of Petri nets, we demonstrate that trace boundedness is ExpSpace-hard (matching the ExpSpace upper bound from [23]), but that the size of the associated bounded expression can be non-primitive-recursive.

Beyond coverability, and as further evidence to the interest of trace boundedness for the verification of WSTS, we show that all ω-regular word properties can be checked against the set of infinite traces of trace bounded ∞-effective cd-WSTS, resulting in a non-trivial recursive class of WSTS with decidable liveness (Section 6.2). Liveness properties are in general undecidable in cd-WSTS [24,25]: techniques for propositional linear-time temporal logic (LTL) model checking are not guaranteed to terminate [26,27] or limited to subclasses, like Petri nets [28]. As a consequence of our result, action-based (aka transition-based) LTL model checking is decidable for cd-WSTS (Section 6.3), whereas state-based properties are undecidable for trace bounded cd-WSTS [29].

One might fear that trace boundedness is too strong a property to be of any practical use. For instance, commutations, as created by concurrent transitions, often result in trace unboundedness. However, bear in mind that the same issues more broadly affect all forward analysis techniques, and have been alleviated in tools through various heuristics. Trace boundedness offers a new insight into why such heuristics work, and can be used as a theoretical foundation for their principled development; we illustrate this point in Section 7 where we introduce trace boundedness modulo a partial commutation relation. We demonstrate the interest of this extension by verifying a liveness property on the Alternating Bit Protocol with a bounded number of sessions.

This work results in an array of concrete classes of WSTS, including lossy channel systems [10], broadcast protocols [26], and Petri nets and their monotone extensions, such as reset/transfer Petri nets [30], for which trace boundedness is decidable and implies both computability of the full coverability set and decidability of liveness properties. Even for trace unbounded systems, it provides a new foundation for the heuristics currently employed by tools to help termination, as with the commutation reductions we just mentioned.

2. Background

2.1. A running example

We consider throughout this paper an example (see Fig. 1) inspired by the recent literature on *asynchronous* or *event-based* programming [31,32], namely that of a client performing n asynchronous remote procedure calls (corresponding to the post(r,rpc) statement on line 7), of which at most P can simultaneously be pending. Such piped—or windowed—clients are commonly employed to prevent server saturation.

The abstracted "producer/consumer" Petri net for this program (ignoring the grayed parts for now) has two transitions i and e modeling the **if** and **else** branches of lines 6 and 9 respectively. The deterministic choice between these two branches

```
1   // Performs n invocations of the rpc() function
2   // with at most P>=1 simultaneous concurrent calls
3   piped_multirpc (int n) {
4   int sent = n, recv = n; rendezvous rdv;
5   while (recv > 0)
6   if (sent > 0 && recv − sent < P) {
7   post(rdv, rpc); // asynchronous call
8   sent−−;
9   } else { // sent == 0 || recv − sent >= P
10  wait(rdv); // a rpc has returned
11  recv−−;
12  }
13  }
```

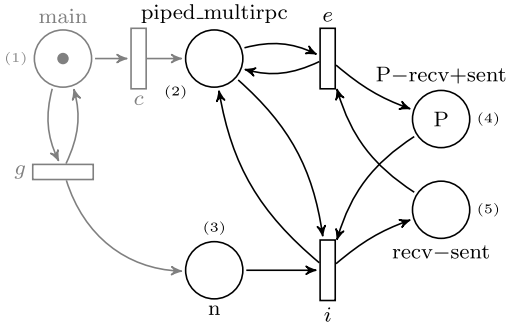

Fig. 1. A piped RPC client in C-like syntax, and its Petri net modelization.

is here replaced by a nondeterministic one, where the program can choose the **else** branch and wait for some rpc call to return before the window of pending calls is exhausted. Observe that we can recover the original program behavior by further controlling the Petri net with the *bounded* regular language $i^P(ei)^*e^P$ (P is fixed), i.e. taking the intersection by synchronous product with a deterministic finite automaton for $i^P(ei)^*e^P$. This is an example of a trace bounded system.

Even without bounded control, the Petri net of Fig. 1 has a bounded, finite, language for each fixed initial n; however, for $P \geq 2$, if we expand it for parametric verification with the left grayed area to allow any n (or set $n = \omega$ as initial value to switch to server mode), then its language becomes unbounded. We will reuse this example in Section 3 when characterizing unboundedness in cd-WSTS. The full system is of course bounded when synchronized with a deterministic finite automaton for the language $g^*ci^P(ei)^*e^P$.

2.2. Definitions

Languages. Let Σ be a finite alphabet; we denote by Σ^* the set of finite sequences of elements from Σ, and by Σ^ω that of infinite sequences; $\Sigma^\infty \stackrel{\text{def}}{=} \Sigma^* \cup \Sigma^\omega$. We denote the empty sequence by ε, the set of non-empty finite sequences by $\Sigma^+ \stackrel{\text{def}}{=} \Sigma^* \setminus \{\varepsilon\}$, the length of a sequence w by $|w|$, the left quotients of a language $L_2 \subseteq L^\infty$ by a language $L_1 \subseteq \Sigma^*$ by $L_1^{-1}L_2 \stackrel{\text{def}}{=} \{v \in \Sigma^\infty \mid \exists u \in L_1, uv \in L_2\}$, and the set of finite prefixes of L_2 by $\mathsf{Pref}(L_2) \stackrel{\text{def}}{=} \{u \in \Sigma^* \mid \exists v \in \Sigma^\infty, uv \in L\}$.

We make regular use of the closure of bounded languages by finite union, intersection and concatenation, taking subsets, prefixes, suffixes, and factors, and of the following sufficient condition for the unboundedness of a language L [1, Lemma 5.3]: the existence of two words u and v in Σ^+, such that $uv \neq vu$ and each word in $\{u, v\}^*$ is a factor of some word in L.

Orderings. Given a relation R on $A \times B$, we denote by R^{-1} its inverse, by $R(C) \subseteq B$ the image of $C \subseteq A$, by R^* its transitive reflexive closure if $R(A) \subseteq A$, and by $\mathsf{dom}\, R \stackrel{\text{def}}{=} R^{-1}(B)$ its domain.

A *quasi ordering* $\leq$ is a reflexive and transitive relation on a set S. We write $\geq = \leq^{-1}$ for the converse quasi order, $< \stackrel{\text{def}}{=} \leq \setminus \geq$ for the associated strict order, and $\equiv \stackrel{\text{def}}{=} \leq \cap \leq^{-1}$ for the associated equivalence relation. The *$\leq$-upward closure* $\uparrow C$ of a set $C \subseteq S$ is $\{s \in S \mid \exists c \in C, c \leq s\}$; its *$\leq$-downward closure* is $\downarrow C \stackrel{\text{def}}{=} \{s \in S \mid \exists c \in C, c \geq s\}$. A set C is *$\leq$-upward closed* (resp. $\leq$-downward closed) if $\uparrow C = C$ (resp. $\downarrow C = C$). A set B is a *basis* for an upward-closed set C (resp. downward-closed) if $\uparrow B = C$ (resp. $\downarrow B = C$). An upper bound $s \in S$ of a set A verifies $a \leq s$ for all a of A, while we denote its *least upper bound*, if it exists, by $\mathsf{lub}(A)$.

A *well quasi ordering* (wqo) is a quasi ordering such that for any infinite sequence $s_0 s_1 s_2 \cdots$ of S^ω there exist $i < j$ in $\mathbb{N}$ such that $s_i \leq s_j$. Equivalently, there does not exist any strictly descending chain $s_0 > s_1 > \cdots > s_i > \cdots$, and any *antichain*, i.e. set of pairwise incomparable elements, is finite. In particular, the set of minimal elements of an upward-closed set C is finite when quotiented by $\equiv$, and is a basis for C. Pointwise comparison $\leq$ in $\mathbb{N}^k$, and scattered subword comparison $\preceq$ on finite sequences in Σ^* are well quasi orders by Higman's Lemma.

Continuous directed complete partial orders. A *directed subset* $D \neq \emptyset$ of S is such that any pair $\{x, y\}$ of elements of D has an upper bound in D. A *directed complete partial order* (dcpo) is such that any directed subset has a least upper bound. A subset O of a dcpo is *open* if it is upward-closed and if, for any directed subset D such that $\mathsf{lub}(D)$ is in O, $D \cap O \neq \emptyset$. A partial function f on a dcpo is *partial continuous* if it is monotonic, $\mathsf{dom} f$ is open, and for any directed subset D of $\mathsf{dom} f$, $\mathsf{lub}(f(D)) = f(\mathsf{lub}(D))$. Two elements s and s' of a dcpo are in a *way below* relation, noted $s \ll s'$, if for every directed subset D such that $\mathsf{lub}(D) \leq s'$, there exists $s'' \in D$ s.t. $s \leq s''$. A dcpo is *continuous* if, for every s' in S, $\mathsf{wb}(s') \stackrel{\text{def}}{=} \{s \in S \mid s \ll s'\}$ is directed and has s' as least upper bound.

Well structured transition systems. A *labeled transition system* (LTS) $\mathcal{S} = \langle S, s_0, \Sigma, \rightarrow \rangle$ comprises a set S of states, an initial state $s_0 \in S$, a finite set of labels Σ, a transition relation $\rightarrow$ on S defined as the union of the relations $\xrightarrow{a} \subseteq S \times S$ for each a in Σ. The relations are extended to sequences in Σ^* by $s \xrightarrow{\varepsilon} s$ and $s \xrightarrow{aw} s''$ for a in Σ and w in Σ^* if there exists s' in S such that $s \xrightarrow{a} s'$ and $s' \xrightarrow{w} s''$. We write $\mathcal{S}(s)$ for the same LTS with s in S as initial state (instead of s_0). A LTS is

- *uniformly bounded branching* if there exists $k \in \mathbb{N}$ such that $\mathsf{Post}_{\mathcal{S}}(s) \stackrel{\text{def}}{=} \{s' \in S \mid s \rightarrow s'\}$ contains less than k elements for all s in S,
- *deterministic* if $\xrightarrow{a}$ is a partial function for each a in Σ—and is thus uniformly bounded branching; we abuse notation in this case and identify u with the partial function $\xrightarrow{u}$ for u in Σ^*,
- *state bounded* if its *reachability set* $\mathsf{Post}^*_{\mathcal{S}}(s_0) \stackrel{\text{def}}{=} \{s \in S \mid s_0 \rightarrow^* s\}$ is finite,
- *trace bounded* if its *trace set* $T(\mathcal{S}) \stackrel{\text{def}}{=} \{w \in \Sigma^* \mid \exists s \in S, s_0 \xrightarrow{w} s\}$ is a bounded language,
- *terminating* if its trace set $T(\mathcal{S})$ is finite.

A *well-structured transition system* (WSTS) [17,8,33] $\langle S, s_0, \Sigma, \rightarrow, \leq, F \rangle$ is a labeled transition system $\langle S, s_0, \Sigma, \rightarrow \rangle$ endowed with a wqo $\leq$ on S and an $\leq$-upward closed set of final states F, such that $\rightarrow$ is *monotonic* wrt. $\leq$: for any s_1, s_2, s_3 in S and a in Σ, if $s_1 \leq s_2$ and $s_1 \xrightarrow{a} s_3$, then there exists $s_4 \geq s_3$ in S with $s_2 \xrightarrow{a} s_4$.

The *language* of a WSTS is defined as $L(\mathcal{S}) \stackrel{\text{def}}{=} \{w \in \Sigma^* \mid \exists s \in F, s_0 \xrightarrow{w} s\}$; see Geeraerts et al. [34] for a general study of such languages. In the context of Petri nets, $L(\mathcal{S})$ is also called the *covering* or *weak* language, and $T(\mathcal{S})$ the *prefix* language. Observe that a *deterministic finite-state automaton* (DFA) is a deterministic WSTS $\mathcal{A} = \langle Q, q_0, \Sigma, \delta, =, F \rangle$, where Q is finite (we shall later omit $=$ from the definition of DFAs).

Given two WSTS $\mathcal{S}_1 = \langle S_1, s_{0,1}, \Sigma, \rightarrow_1, \leq_1, F_1 \rangle$ and $\mathcal{S}_2 = \langle S_2, s_{0,2}, \Sigma, \rightarrow_2, \leq_2, F_2 \rangle$, their *synchronous product* $\mathcal{S}_1 \times \mathcal{S}_2 \stackrel{\text{def}}{=} \langle S_1 \times S_2, (s_{0,1}, s_{0,2}), \Sigma, \rightarrow_\times, \leq_\times, F_1 \times F_2 \rangle$, where for all s_1, s_1' in S_1, s_2, s_2' in S_2, a in Σ, $(s_1, s_2) \xrightarrow{a}_\times (s_1', s_2')$ if and only if $s_1 \xrightarrow{a}_1 s_1'$ and $s_2 \xrightarrow{a}_2 s_2'$, and $(s_1, s_2) \leq_\times (s_1', s_2')$ if and only if $s_1 \leq_1 s_1'$ and $s_2 \leq_2 s_2'$, is again a WSTS, such that $L(\mathcal{S}_1 \times \mathcal{S}_2) = L(\mathcal{S}_1) \cap L(\mathcal{S}_2)$.

We often consider the case $F = S$ and omit F from the WSTS definition, as we are more interested in trace sets, which provide more evidence on the reachability sets.

Coverability. A WSTS is Pred-*effective* if $\rightarrow$ and $\leq$ are decidable, and a finite basis for $\uparrow \mathsf{Pred}_{\mathcal{S}}(\uparrow s, a) \stackrel{\text{def}}{=} \uparrow \{s' \in S \mid \exists s'' \in S, s' \xrightarrow{a} s'' \text{ and } s \leq s''\}$ can effectively be computed for all s in S and a in Σ [33].

The *cover set* of a WSTS is $\mathsf{Cover}_{\mathcal{S}}(s_0) \stackrel{\text{def}}{=} \downarrow \mathsf{Post}^*_{\mathcal{S}}(s_0)$, and it is decidable whether a given state s belongs to $\mathsf{Cover}_{\mathcal{S}}(s_0)$ for finite branching Pred-effective WSTS, thanks to a backward algorithm that checks whether s_0 belongs to $\uparrow \mathsf{Pred}^*_{\mathcal{S}}(\uparrow s) \stackrel{\text{def}}{=} \uparrow \{s' \in S \mid \exists s'' \in S, s' \rightarrow^* s'' \text{ and } s'' \geq s\}$. One can also decide the emptiness of the language of a WSTS, by checking whether s_0 belongs to $\uparrow \mathsf{Pred}^*_{\mathcal{S}}(F)$.

Flattenings. Let $\mathcal{A}$ be a DFA with a bounded language. The synchronous product $\mathcal{S} \times \mathcal{A}$ of $\mathcal{S}$ and $\mathcal{A}$ is a *flattening* of $\mathcal{S}$. Consider the projection π from $S \times Q$ to S defined by $\pi(s, q) \stackrel{\text{def}}{=} s$; then $\mathcal{S}$ is *post** *flattable* if there exists a flattening $\mathcal{S}'$ of $\mathcal{S}$ such that $\mathsf{Post}^*_{\mathcal{S}}(s_0) = \pi(\mathsf{Post}^*_{\mathcal{S}'}((s_0, q_0)))$. In the same way, it is *cover flattable* if $\mathsf{Cover}_{\mathcal{S}}(s_0) = \pi(\mathsf{Cover}_{\mathcal{S}'}((s_0, q_0)))$, and *trace flattable* if $T(\mathcal{S}) = T(\mathcal{S}')$. Remark that

1. trace flattability is equivalent to the boundedness of the trace set, and that
2. trace flattability implies post* flattability, which in turn implies cover flattability.

Complete WSTS. A deterministic WSTS $\langle S, s_0, \Sigma, \rightarrow, \leq \rangle$ is *complete* (a cd-WSTS) if $(S, \leq)$ is a continuous dcpo and each transition function a for a in Σ is partial continuous [14,2]. The *lub-acceleration* u^ω of a partial continuous function u on S, u in Σ^+, is again a partial function on S defined by

$$\mathsf{dom}\, u^\omega \stackrel{\text{def}}{=} \{s \in \mathsf{dom}\, u \mid s \leq u(s)\}$$

$$u^\omega(s) \stackrel{\text{def}}{=} \mathsf{lub}(\{u^n(s) \mid n \in \mathbb{N}\}) \qquad \text{for all } s \text{ in } \mathsf{dom}\, u^\omega.$$

A complete WSTS is ∞-*effective* if u^ω is computable for every u in Σ^+.

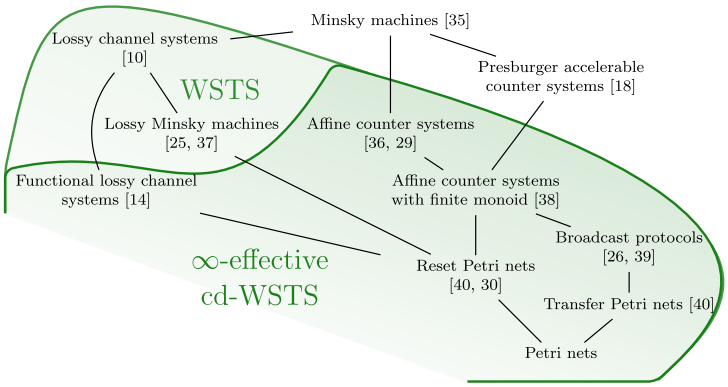

Fig. 2. Classes of systems mentioned in the paper, with a few relevant references (see [10,14,18,25,26,29,30,35–40]).

2.3. Working hypotheses

Our decidability results rely on some effectiveness assumptions for a restricted class of WSTS: the complete deterministic ones. We discuss in this section the exact scope of these hypotheses. As an appetizer, notice that both trace boundedness and action-based ω-regular properties are only concerned with trace sets, hence one can more generally consider classes of WSTS for which a trace-equivalent complete deterministic system can effectively be found. Fig. 2 presents the various classes of systems mentioned at one point or another in the main text or in the proofs. It also provides a good way to emphasize the applicability of our results on ∞-effective cd-WSTS.

Completeness. Finkel and Goubault-Larrecq [2] define *ω^2-WSTS* as the class of systems that can be completed, and provide an extensive off-the-shelf algebra of datatypes with their completions [14]. As they argue, all the concrete classes of deterministic WSTS considered in the literature are ω^2. Completed systems share their sets of finite and infinite traces with the original systems: the added limit states only influence transfinite sequences of transitions.

For instance, the whole class of *affine counter systems*, with affine transition functions of form $f(\boldsymbol{x}) = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{b}$, with $\boldsymbol{A}$ a $k \times k$ matrix of non-negative integers and $\boldsymbol{b}$ a vector of k integers—encompassing reset/transfer Petri nets and broadcast protocols—can be completed to configurations in $(\mathbb{N} \cup \{\omega\})^k$. Similarly, *functional lossy channel systems*—a deterministic variant of lossy channel systems [14, see also Section 4.3]—can work on *products* [27, Corollary 6.5]. On both accounts, the completed functions are partial continuous.

Determinism. Beyond deterministic systems, one can consider *finite branching* WSTS [14]. These are defined as deterministic WSTS equipped with a labeling function σ. Consider a deterministic WSTS $\langle S, s_0, \mathcal{F}, \to, \leq \rangle$, where $\mathcal{F}$ is a finite alphabet of action names; together with a labeling $\sigma : \mathcal{F} \to \Sigma$, it defines a possibly non-deterministic WSTS $\langle S, s_0, \Sigma, \to', \leq \rangle$ with $s \xrightarrow{a}{}' s'$ if and only if there exists f in $\mathcal{F}$ such that $s \xrightarrow{f} s'$ and $\sigma(f) = a$.

Assuming basic effectiveness assumptions on the so-called *principal filters* $\uparrow s$ of S, we can decide the following sufficient condition for determinism on finite branching WSTS:

Proposition 1. *Let $\mathcal{S}$ be a WSTS defined by a deterministic WSTS $\langle S, s_0, \mathcal{F}, \to, \leq \rangle$ along with a labeling $\sigma : \mathcal{F} \to \Sigma$. If finite bases can be computed for $\uparrow s \cap \uparrow s'$ for all s, s' in S, and for S itself, then one can decide whether, for all reachable states s of S and pairs (f, f') of transition functions in $\mathcal{F}$ with $\sigma(f) = \sigma(f')$, $s \in \mathsf{dom} \xrightarrow{f} \cap \mathsf{dom} \xrightarrow{f'}$ implies $f = f'$.*

Proof. Let B be a finite basis for S, i.e. $\uparrow B = S$, and let $D \stackrel{\text{def}}{=} \mathsf{dom} \xrightarrow{f} \cap \mathsf{dom} \xrightarrow{f'}$.

We can reformulate the existence of an s violating the condition of the proposition as a coverability problem, by checking whether s_0 belongs to $\mathsf{Pred}^*(D)$, which is decidable thanks to the usual backward reachability algorithm if we provide a finite basis for D. To that end, we first compute B_f and $B_{f'}$ two finite bases for $\mathsf{dom} \xrightarrow{f} = \uparrow \mathsf{Pred}_{\mathcal{S}}(\uparrow B, f)$ and $\mathsf{dom} \xrightarrow{f'} = \uparrow \mathsf{Pred}_{\mathcal{S}}(\uparrow B, f)$ using the Pred-effectiveness of $\mathcal{S}$. We then compute a finite basis for

$$D = \bigcup_{s_f \in B_f, s_{f'} \in B_{f'}} (\uparrow s_f \cap \uparrow s_{f'}) \tag{1}$$

using the computation of finite bases for intersections of principal filters. □

For instance, labeled functional lossy channel systems and labeled affine counter systems fit Proposition 1; also note that determinism is known to be ExpSpace-complete for labeled Petri nets [41].

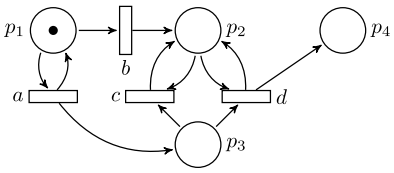

Fig. 3. The Petri net $\mathcal{N}'(1,0,0,0)$, with an unbounded trace set.

Another extension beyond cd-WSTS is possible: Call a system $\mathcal{S}$ *essentially deterministic* if, analogously to the *essentially finite branching* systems of Abdulla et al. [8], for each state s and symbol a, there is a single maximal element inside $\mathsf{Post}_{\mathcal{S}}(s,a) = \{s' \in S \mid s \xrightarrow{a} s'\}$, which we can effectively compute. Indeed, from $\mathcal{S}$ we can construct a deterministic system $\mathcal{S}_d$ with transitions $s \xrightarrow{a} \mathsf{max}(\mathsf{Post}_{\mathcal{S}}(s,a))$ defined whenever $\mathsf{Post}_{\mathcal{S}}(s,a)$ is not empty, for all s in S and a in Σ. Thanks to monotonicity, any string recognized from some state in $\mathsf{Post}_{\mathcal{S}}(s,a)$ can also be recognized from $\mathsf{max}(\mathsf{Post}_{\mathcal{S}}(s,a))$, which entails $T(\mathcal{S}) = T(\mathcal{S}_d)$.

Recall that though most of *infinite branching WSTS* can be embedded into their finite branching WSTS completion [13], this completion has no reason to be uniformly bounded or deterministic.

Finally, one can try to devise trace- and cover-equivalent deterministic semantics for systems with unbounded *but finite* branching, like *functional lossy channel systems* [14] for lossy channel systems, or reset Petri nets for lossy Minsky machines. From a verification standpoint, the deterministic semantics is then equivalent to the classical one.

Effectiveness. All the concrete classes of WSTS we have mentioned are Pred-effective, and we assume this property from all our systems from now on. It also turns out that ∞-effective systems abound, including once more (completed) affine counter systems [2] and functional lossy channel systems.

3. Deciding trace boundedness

We present in this section two semi-algorithms, first for trace boundedness, which relies on the decidability of language emptiness in WSTS, and then for trace unboundedness, for which we show that a finite witness can be found in cd-WSTS. In fact, this second semi-algorithm can be turned into a full-fledged algorithm when some extra care is taken in the search for a witness.

Theorem 2. *Trace boundedness is decidable for ∞-effective cd-WSTS. If the trace set is bounded, then one can compute an adequate bounded expression $w_1^* \cdots w_n^*$ for it.*

An additional remark is that Theorem 2 holds more generally for the boundedness of the *language* $L(\mathcal{S})$ of a WSTS instead of its trace set $T(\mathcal{S})$. Indeed, the semi-algorithm for boundedness would work just as well with $L(\mathcal{S})$, while the semi-algorithm for unboundedness can restrict its search for a witness to $\mathsf{Pre}^*(F)$.

3.1. Trace boundedness

Trace boundedness is semi decidable with a rather straightforward procedure for any WSTS $\mathcal{S}$ (neither completeness nor determinism are necessary): enumerate the possible bounded expressions $w_1^* \cdots w_n^*$ and check whether the trace set $T(\mathcal{S})$ of the WSTS is included in their language. This last operation can be performed by checking the emptiness of the language of the WSTS obtained as the synchronous product $\mathcal{S} \times \mathcal{A}$ of the original system with a DFA $\mathcal{A}$ for the complement of the language of $w_1^* \cdots w_n^*$. If $L(\mathcal{S} \times \mathcal{A})$ is empty, which is decidable thanks to the generic backwards algorithm for WSTS, then we have found a bounded expression for $T(\mathcal{S})$.

3.2. Trace unboundedness

We detail the procedure for trace unboundedness of the trace set. Our construction relies on the existence of a witness of trace unboundedness, which can be found after a finite time in a cd-WSTS by exploring its states using accelerated sequences.

Overview. Let us consider the Petri net $\mathcal{N}'$ with initial marking $(1,0,0,0)$, depicted in Fig. 3, with trace set

$$T(\mathcal{N}'(1,0,0,0)) = a^* \cup \bigcup_{n \geq 0} a^n b \{c,d\}^{\leq n}.$$

Notice that the trace set of $\mathcal{N}'$ with initial marking $(0,1,n,0)$ is bounded for each n: it is $\{c,d\}^{\leq n}$, a finite language. The trace unboundedness of $\mathcal{N}'(1,0,0,0)$ originates in its ability to reach every $(0,1,n,0)$ marking after a sequence of n transitions on a followed by a b transition.

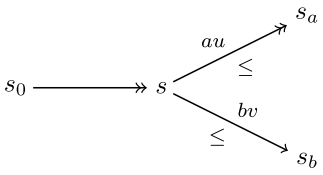

Fig. 4. An increasing fork witnesses trace unboundedness.

Consider now transitions $(1,0,0,0) \xrightarrow{a} (1,0,1,0)$ and $(1,0,0,0) \xrightarrow{b} (0,1,0,0)$. The two systems $\mathcal{N}'(1,0,1,0)$ and $\mathcal{N}'(0,1,0,0)$ are respectively trace unbounded and trace bounded. More generally, Lemma 8 will show later that, if $L \subseteq \Sigma^*$ is an unbounded language, then there exists a in Σ such that $a^{-1}L$ is also unbounded. By repeated applications of, we can find words w of any length $|w| = n$ such that $w^{-1}L$ is still unbounded: this is the case of a^n in our example. This process continues to the infinite, but in a WSTS we will eventually find two states $s_i \leq s_j$, met after $i < j$ steps respectively. Let $s_i \xrightarrow{u} s_j$; by monotonicity we can recognize u^* starting from s_i. In a cd-WSTS, there is a lub-accelerated state s with $s_i \xrightarrow{u^\omega} s$ that represents the effect of all these u transitions; here $(1,0,0,0) \xrightarrow{a^\omega} (1,0,\omega,0)$. The interesting point is that our lub-acceleration finds the correct residual trace set: $T(\mathcal{N}'(1,0,\omega,0)) = (a^*)^{-1}T(\mathcal{N}'(1,0,0,0))$.

Again, we can repeatedly remove accelerated strings from the prefixes of our trace set and keep it unbounded. However, due to the wqo, an infinite succession of lub-accelerations allows us to nest some loops after a finite number of steps. Still with the same example, we reach $(1,0,\omega,0) \xrightarrow{b} (0,1,\omega,0)$, and—thanks to the lub-acceleration—the source of trace unboundedness is now visible because both $(0,1,\omega,0) \xrightarrow{c} (0,1,\omega,0)$ and $(0,1,\omega,0) \xrightarrow{d} (0,1,\omega,1)$ are increasing, thus by monotonicity $T(\mathcal{N}'(0,1,\omega,0)) = \{c,d\}^*$. By continuity, for each string u in $\{c,d\}^*$, there exists n in $\mathbb{N}$ such that $a^n bu$ is an actual trace of $\mathcal{N}'(1,0,0,0)$.

The same reasoning can be applied to the Petri net of Fig. 1 with initial marking $(1,0,0,P,0)$ for $P \geq 2$. As mentioned in Section 2, its trace set is unbounded, but the trace set of $\mathcal{N}$ with initial marking $(0,1,n,P,0)$ is bounded for each n, since it is a finite language. We reach $(1,0,0,P,0) \xrightarrow{g^\omega} (1,0,\omega,P,0) \xrightarrow{ci} (0,1,\omega,P-1,1)$ and see that both $(0,1,\omega,P-1,1) \xrightarrow{ei} (0,1,\omega,P-1,1)$ and $(0,1,\omega,P-1,1) \xrightarrow{ieei} (0,1,\omega,P-1,1)$ are increasing, thus by monotonicity $T(\mathcal{N}(0,1,\omega,P-1,1))$ contains $\{ei, ieei\}^*$. Here continuity comes into play to show that these limit behaviors are reflected in the set of finite traces of the system: in our example, for each string u in $\{ei, ieei\}^*$, there exists a finite n in $\mathbb{N}$ such that $g^n ciu$ is an actual trace of $\mathcal{N}(1,0,0,P,0)$.

Increasing forks. We call the previous witness of trace unboundedness an *increasing fork*, as depicted in schematic form in Fig. 4. Let us first define accelerated runs and languages for complete WSTS, where lub-accelerations are employed.

Definition 3. Let $\mathcal{S} = \langle S, s_0, \Sigma, \to, \leq, F\rangle$ be a cd-WSTS. An *accelerated run* is a finite sequence $\sigma = s_0 s_1 s_2 \cdots s_n$ in S^* such that for all $i \geq 0$, either there exists a in Σ such that

$$s_i \xrightarrow{a} s_{i+1} \qquad \text{(single step)}$$

or there exists u in Σ^+ such that

$$s_i \xrightarrow{u^\omega} s_{i+1}\,. \qquad \text{(accelerated step)}$$

We denote the relation over S defined by such an accelerated run by $s_0 \twoheadrightarrow s_n$. An accelerated run is *accepting* if s_n is in F. The *accelerated language* (resp. *accelerated trace set*) $L_{\text{acc}}(\mathcal{S})$ (resp. $T_{\text{acc}}(\mathcal{S})$) of $\mathcal{S}$ is the set of sequences that label some accepting accelerated run (resp. some accelerated run).

We denote by Σ^*_{acc} the set of finite sequences mixing letters a from Σ and accelerations u^ω where u is a finite sequence from Σ^+; in particular $L_{\text{acc}}(\mathcal{S}) \subseteq \Sigma^*_{\text{acc}}$.

Definition 4. A cd-WSTS $\mathcal{S} = \langle S, s_0, \Sigma, \to, \leq\rangle$ has an *increasing fork* if there exist $a \neq b$ in Σ, u in Σ^*_{acc}, v in Σ^*, and s, $s_a \geq s$, $s_b \geq s$ in S such that $s_0 \twoheadrightarrow s$, $s \xrightarrow{au} s_a$, and $s \xrightarrow{bv} s_b$.

As shown in the following proposition, a semi-algorithm for trace unboundedness in ∞-effective cd-WSTS then consists in an exhaustive search for an increasing fork, by applying non-nested lub-accelerations whenever possible. In fact, by choosing which acceleration sequences to employ in the search for an increasing fork, we can turn this semi-algorithm into a full algorithm; we will see this in more detail in Section 5.2.

Proposition 5. *A cd-WSTS has an unbounded trace set if and only if it has an increasing fork.*

The remainder of the section details the proof of Proposition 5.

An increasing fork implies unboundedness. The following lemma shows that, thanks to continuity, what happens in accelerated runs is mirrored in finite runs.

Lemma 6. *Let $\mathcal{S}$ be a cd-WSTS and $n \geq 0$. If*

$$w_n = v_{n+1} u_n^\omega v_n \cdots u_1^\omega v_1 \in T_{\mathsf{acc}}(\mathcal{S})$$

with the u_i in Σ^+ and the v_i in Σ^*, then there exist $k_1, \ldots, k_n$ in $\mathbb{N}$, such that

$$w'_n = v_{n+1} u_n^{k_n} v_n \cdots u_1^{k_1} v_1 \in T(\mathcal{S}) \,.$$

Proof. We proceed by induction on n. In the base case where $n = 0$, $w_0 = v_1$ belongs trivially to $T(\mathcal{S})$—this concludes the proof if we are considering words in $T(\mathcal{S})$. For the induction part, let s be a state such that

$$s_0 \xrightarrow{v_{n+1} u_n^\omega} s \xrightarrow{v_n u_{n-1}^\omega v_{n-1} \cdots u_1^\omega v_1} s_f \,,$$

i.e. $w_{n-1} = v_n u_{n-1}^\omega v_{n-1} \cdots u_1^\omega v_1$ is in $T_{\mathsf{acc}}(\mathcal{S}(s))$. Therefore, using the induction hypothesis, we can find $k_1, \ldots, k_{n-1}$ in $\mathbb{N}$ such that

$$w'_{n-1} = v_n u_{n-1}^{k_{n-1}} v_{n-1} \cdots u_1^{k_1} v_1 \in T(\mathcal{S}(s)) \,.$$

Because $\mathcal{S}$ is complete, $\xrightarrow{w'_{n-1}}$ is a partial continuous function, hence with an open domain O. This domain O contains in particular s, which by definition of u_n^ω is the lub of the directed set $\{s' \mid \exists m \in \mathbb{N}, s_0 \xrightarrow{v_{n+1} u_n^m} s'\}$. By definition of an open set, there exists an element s' in $\{s' \mid \exists m \in \mathbb{N}, s_0 \xrightarrow{v_{n+1} u_n^m} s'\} \cap O$, i.e. there exists k_n in $\mathbb{N}$ s.t. $s_0 \xrightarrow{v_{n+1} u_n^{k_n}} s'$ and s' can fire the transition sequence w'_{n-1}. □

Continuity is crucial for the soundness of our procedure, as can be better understood by considering the example of the WSTS $\mathcal{S}' = \langle \mathbb{N} \uplus \{\omega\}, 0, \{a, b\}, \rightarrow, \leq \rangle$ with transitions

$$\forall n \in \mathbb{N}, n \xrightarrow{a} n+1, \qquad \omega \xrightarrow{a} \omega, \qquad \omega \xrightarrow{b} \omega \,.$$

We obtain a bounded set of finite traces $T(\mathcal{S}'(0)) = a^*$, but reach the configuration ω through lub-accelerations, and then find an increasing fork with $T(\mathcal{S}'(\omega)) = \{a, b\}^*$, an unbounded language. Observe that $\mathbb{N}$ is a directed set with ω as lub, thus the domain of $\xrightarrow{b}$ should contain some elements of $\mathbb{N}$ in order to be open: $\mathcal{S}'$ is not a complete WSTS.

Lemma 7. *Let $\mathcal{S}$ be a cd-WSTS. If $\mathcal{S}$ has an increasing fork, then $T(\mathcal{S})$ is unbounded.*

Proof. Suppose that $\mathcal{S}$ has an increasing fork with the same notations as in Definition 4, and let w in Σ_{acc}^* be such that $s_0 \xrightarrow{w} s$. By monotonicity, we can fire from s the accelerated transitions of au and the transitions of bv in any order and any number of time, hence

$$w\{au, bv\}^* \subseteq T_{\mathsf{acc}}(\mathcal{S}) \,.$$

Suppose now that $T(\mathcal{S})$ is bounded, i.e. that there exists $w_1, \ldots, w_n$ such that $T(\mathcal{S}) \subseteq w_1^* \cdots w_n^*$. Then, there exists a DFA $\mathcal{A} = \langle Q, q_0, \Sigma, \delta, F \rangle$ such that $L(\mathcal{A}) = w_1^* \cdots w_n^*$ and thus $T(\mathcal{S}) \subseteq L(\mathcal{A})$. Set $N = |Q| + 1$. We have in particular

$$w(bv)^N au(bv)^N au \cdots au(bv)^N \in T_{\mathsf{acc}}(\mathcal{S})$$

with N repetitions of the $(bv)^N$ factor. By Lemma 6, we can find some adequate finite sequences $w', u_1, \ldots, u_{N-1}$ in Σ^* such that

$$w'(bv)^N au_1 (bv)^N au_2 \cdots au_{N-1} (bv)^N \in T(\mathcal{S}) \,.$$

Because $T(\mathcal{S}) \subseteq L(\mathcal{A})$, this word is also accepted by $\mathcal{A}$, and we can find an accepting run for it. Since $N = |Q| + 1$, for each of the N occurrences of the $(bv)^N$ factor, there exists a state q_i in Q such that $\delta(q_i, (bv)^{k_i}) = q_i$ for some $k_i > 0$. Thus the accepting run in $\mathcal{A}$ is of form

$$q_0 \xrightarrow{w'(bv)^{N-k_1-k'_1}} q_1 \xrightarrow{(bv)^{k_1}} q_1 \xrightarrow{(bv)^{k'_1} au_1 (bv)^{N-k_2-k'_2}} q_2,$$

$$q_2 \xrightarrow{(bv)^{k_2}} q_2 \xrightarrow{(bv)^{k'_2} au_2 \cdots au_{N-1} (bv)^{N-k_N-k'_N}} q_N,$$

$$q_N \xrightarrow{(bv)^{k_N}} q_N \xrightarrow{(bv)^{k'_N}} q_f \in F$$

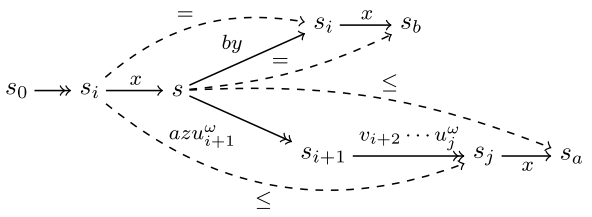

Fig. 5. The construction of an increasing fork in the proof of Lemma 11.

for some integers $k'_i \geq 0$. Again, since $N = |Q| + 1$, there exist $1 \leq i < j \leq N$ such that $q_i = q_j$, hence

$$\delta(q_i, (bv)^{k'_i} a u_i \cdots a u_{j-1} (bv)^{N-k_j-k'_j}) = q_i \,.$$

This implies that $\{(bv)^{k_i}, (bv)^{k'_i} a u_i \cdots a u_{j-1} (bv)^{N-k_j-k'_j}\}^*$ is contained in the set of factors of $L(\mathcal{A})$ with

$$(bv)^{k_i+k'_i} a u_i \cdots a u_{j-1} (bv)^{N-k_j-k'_j} \neq (bv)^{k'_i} a u_i \cdots a u_{j-1} (bv)^{N-k_j-k'_j+k_i}$$

since $a \neq b$, thus $L(\mathcal{A})$ is an unbounded language [1, Lemma 5.3], a contradiction. □

Unboundedness implies an increasing fork. We follow the arguments presented on the example of Fig. 1, and prove that an increasing fork can always be found in an unbounded cd-WSTS.

Lemma 8. *Let $L \subseteq \Sigma^*$ be an unbounded language. There exists a in Σ such that $a^{-1}L$ is also unbounded.*

Proof. Observe that $L = \bigcup_{a\in\Sigma} a \cdot (a^{-1}L)$. If every $a^{-1}L$ were bounded, since bounded languages are closed by finite union and concatenation, L would also be bounded. □

Definition 9. Let be $L \subseteq \Sigma^*$ and $w \in \Sigma^+$. The *removal* of w from L is the language $\overline{w}L = ((w^*)^{-1}L) \setminus w\Sigma^*$.

Lemma 10. *If a cd-WSTS $\mathcal{S}$ has an unbounded trace set $T(\mathcal{S})$ in Σ^*, and L is an unbounded language with $L \subseteq T(\mathcal{S})$ then there are two words v in Σ^* and u in Σ^+ such that $vu^{\omega} \in T_{\mathsf{acc}}(\mathcal{S})$, $vu \in \mathsf{Pref}(L)$ and $\overline{u}(v^{-1}L)$ is also unbounded.*

Proof. By Lemma 8 we can find a sequence $(a_i)_{i>0} \in \Sigma^{\omega}$ such that for all n in $\mathbb{N}$, $(a_1 \cdots a_n)^{-1}L$ is unbounded. Let $(s_i)_{i\geq 0}$ be the corresponding sequence of configurations in S^{ω}, such that $s_i \xrightarrow{a_{i+1}} s_{i+1}$. Because $(S, \leq)$ is a wqo, there exist $i < j$ such that $s_i \leq s_j$. We set $v = a_1 \cdots a_i$ and $u = a_{i+1} \cdots a_j$, which gives us $v \cdot u^{\omega} \in T_{\mathsf{acc}}(\mathcal{S})$. Remark that $v^{-1}L$ is unbounded, and, since $u^*\overline{u}(v^{-1}L) = u^*(v^{-1}L)$, $\overline{u}(v^{-1}L)$ is unbounded too. □

Note that it is also possible to ask that $|vu| \geq n$ for any given n, which we do in the proof of the following lemma.

Lemma 11. *If a cd-WSTS has an unbounded trace set, then it has an increasing fork.*

Proof. We define simultaneously three infinite sequences, $(v_i, u_i)_{i>0}$ of pairs of words in $\Sigma^* \times \Sigma^+$, $(L_i)_{i\geq 0}$ of unbounded languages, and $(s_i)_{i\geq 0}$ of initial configurations: let $L_0 \stackrel{\text{def}}{=} T(\mathcal{S})$ and s_0 the initial configuration of $\mathcal{S}$, and

- v_{i+1}, u_{i+1} are chosen using Lemma 10 such that $v_{i+1}u_{i+1}^{\omega}$ is in $T_{\mathsf{acc}}(\mathcal{S}(s_i))$, $v_{i+1}u_{i+1}$ is in $\mathsf{Pref}(L_i)$, $|v_{i+1} \cdot u_{i+1}| \geq |u_i|$ if $i > 0$, and $\overline{u_{i+1}}(v_{i+1}^{-1}L_i)$ is unbounded;
- $s_i \xrightarrow{v_{i+1}u_{i+1}^{\omega}} s_{i+1}$;
- $L_{i+1} \stackrel{\text{def}}{=} \overline{u_{i+1}}(v_{i+1}^{-1}L_i)$.

Since $\overline{u_{i+1}}(v_{i+1}^{-1}L_i) \subseteq T(\mathcal{S}(s_{i+1}))$, we can effectively iterate the construction by the last point above.

Due to the wqo, there exist $i < j$ such that $s_i \leq s_j$. By construction u_i is not a prefix of $v_{i+1}u_{i+1}$ and $|v_{i+1}u_{i+1}| \geq |u_i|$, so there exist $a \neq b$ in Σ and a longest common prefix x in Σ^* such that $u_i = xby$ and $v_{i+1}u_{i+1} = xaz$ for some y, z in Σ^*.

We exhibit an increasing fork by selecting s, s_a, s_b such that (see Fig. 5):

$$s_i \xrightarrow{x} s \qquad s \xrightarrow{azu_{i+1}^{\omega} v_{i+2} u_{i+2}^{\omega} \cdots v_j u_j^{\omega} x} s_a \qquad s \xrightarrow{byx} s_b \,. \qquad \square$$

We will refine the arguments of Lemma 11 in Section 5.2. In particular, note that a strategy where the $v_{i+1}u_{i+1}$ sequences are the shortest possible defines a means to perform an *exhaustive* search for this particular brand of increasing forks, this at no loss of generality as far as trace boundedness is concerned. Thus our semi-algorithm is actually an algorithm.

4. Undecidable cases

This section establishes that the decidability of the trace boundedness property for cd-WSTS disappears if we consider more general systems or a more general property. Unsurprisingly, trace boundedness is undecidable on general systems like 2-counter Minsky machines (Section 4.1). It also becomes undecidable if we relax the determinism condition, as shown by considering the case of labeled reset Petri nets (Section 4.2). We conclude by proving that post* flattability is undecidable for deterministic WSTS (Section 4.3). Note that completeness is irrelevant in all the following reductions.

4.1. General systems

We demonstrate that the trace boundedness problem is undecidable for deterministic Minsky machines, by reduction from their halting problem. We could rely on Rice's Theorem, but find it more enlightening to present a direct proof that turns a Minsky machine $\mathcal{M}$ into a new one $\mathcal{M}'$, which halts if and only if $\mathcal{M}$ halts. The new machine has a bounded trace set if it halts, and an unbounded trace set otherwise.

Let us first recall that a deterministic *Minsky machine* is a tuple $\mathcal{M} = \langle Q, \delta, C, q_0 \rangle$ where Q is a finite set of labels, δ a finite set of actions, C a finite set of counters that take their values in $\mathbb{N}$, and $q_0 \in Q$ an initial label. A label q identifies a unique action in δ, which is of one of the following three forms:

q : `if` $c = 0$ `goto` q' `else` c--; `goto` q''

q : c++; `goto` q'

q : `halt`

where q' and q'' are labels and c is a counter. A configuration of $\mathcal{M}$ is a pair (q, m) with q a label in Q and m a marking in $\mathbb{N}^C$, and leads to a single next configuration (q', m') by applying the action labeled by q—which should be self-explaining—if different from `halt`. A run of $\mathcal{M}$ starts with configuration $(q_0, \mathbf{0})$ and halts if it reaches a configuration that labels a `halt` action. We define the corresponding LTS semantics by $(q, m) \xrightarrow{q} (q', m')$ if (q, m) and (q', m') are two successive configurations of $\mathcal{M}$; note that there is at most one possible transition from any (q, m) configuration, thus this LTS is deterministic. It is undecidable whether a 2-counter Minsky machine halts [35].

We also need a small technical lemma that relates the size of bounded expressions with the size of some special words.

Definition 12. The *size* of a bounded expression $w_1^* \cdots w_n^*$ is $\sum_{i=1}^{n} |w_i|$.

Lemma 13. *Let $v_m \in (\Sigma \uplus \Delta)^*$ be a word of form $u_1x_1u_2x_2u_3 \dots u_mx_m$ with $m \in \mathbb{N}$, $u_i \in \Sigma^+$, $x_i \in \Delta^+$ and $|x_i| < |x_{i+1}|$ for all i. If there exist $w_1, \dots, w_n$ in $(\Sigma \uplus \Delta)^*$ such that $v_m \in w_1^* \cdots w_n^*$, then $\sum_{i=1}^{n} |w_i| \geq m$.*

Proof. We consider for this proof the number of alphabet alternations $\mathsf{alt}(w)$ of a word w in $(\Sigma \uplus \Delta)^*$, which we define using the unique decomposition of w as $y_1 \cdots y_{\mathsf{alt}(w)}$ where each y_i factor is non-empty and in an alphabet different from that of its successor. For instance, $\mathsf{alt}(v_m)$ is $2m$. We relate the number of alternations produced by words w_i of a bounded expression for v_m with their lengths. More precisely, we show that, if

$$v_m = w_1^{j_1} \cdots w_n^{j_n} ,$$

then for all $1 \leq i \leq n$

$$\mathsf{alt}(w_i^{j_i}) \leq 2|w_i| . \tag{2}$$

Clearly, (2) holds if $|w_i| = 0$ or $j_i = 0$. If a word w_i is in Σ^+ or Δ^+, then $\mathsf{alt}(w_i^j) = 1$ for all $j > 0$ and (2) holds again. Otherwise, the word w_i contains at least one alternation, and then $j_i \leq 2$: otherwise there would be two maximal x factors (in Δ^+) in v_m with the same length. As each alternation inside w_i requires at least one more symbol, we verify (2). Therefore,

$$2m = \mathsf{alt}(w_1^{j_1} \cdots w_n^{j_n}) \leq \sum_{i=1}^{n} \mathsf{alt}(w_i^{j_i}) \leq 2 \sum_{i=1}^{n} |w_i| . \qquad \square$$

Proposition 14. *Trace boundedness is undecidable for 2-counter Minsky machines.*

Proof. We reduce from the halting problem for a 2-counter Minsky machine $\mathcal{M}$ with initial counters at zero. We construct a 4-counter Minksy machine $\mathcal{M}'$ such that $T(\mathcal{M}')$ is bounded if and only if $\mathcal{M}$ halts.

The machine $\mathcal{M}'$ adds two extra counters c_3 and c_4, initially set to zero, and new labels and actions to $\mathcal{M}$. These are used to insert longer and longer sequences of transitions at each step of the original machine: each label q gives rise to the creation of five new labels $q', q'', q^\dagger, q^\ddagger, q^\flat$ that identify the following actions

$$q' : \texttt{if}\ c_3 = 0\ \texttt{goto}\ q^\dagger\ \texttt{else}\ c_3\texttt{--};\ \texttt{goto}\ q''$$
$$q'' : c_4\texttt{++};\ \texttt{goto}\ q'$$
$$q^\dagger : \texttt{if}\ c_4 = 0\ \texttt{goto}\ q^\flat\ \texttt{else}\ c_4\texttt{--};\ \texttt{goto}\ q^\ddagger$$
$$q^\ddagger : c_3\texttt{++};\ \texttt{goto}\ q^\dagger$$
$$q^\flat : c_3\texttt{++};\ \texttt{goto}\ q$$

and each subinstruction $\texttt{goto}\,q$ in the original actions is replaced by $\texttt{goto}\,q'$. The machine $\mathcal{M}'$ halts iff $\mathcal{M}$ halts. If it halts, then its trace set $T(\mathcal{M}')$ is a singleton $\{w\}$, and thus is bounded. If it does not halt, then its trace set is the set of finite prefixes of an infinite trace of form

$$q_0(q_1'q_1'')^0q_1'u_1q_1(q_2'q_2'')^1q_2'u_2q_2(q_3'q_3'')^2q_3'u_3 \cdots q_i(q_{i+1}'q_{i+1}'')^iq_{i+1}'u_{i+1}q_{i+1}\cdots$$

where $q_0q_1q_2\cdots q_iq_{i+1}\cdots$ is the corresponding trace of the execution of $\mathcal{M}$, and the u_j are sequences in $\{q_j^\dagger, q_j^\ddagger, q_j^\flat\}^*$. By Lemma 13, no expression $w_1^*\cdots w_n^*$ of finite size can be such that $T(\mathcal{M}') \subseteq w_1^*\cdots w_n^*$.

We then conclude thanks to the (classical) encoding of our 4-counter machine $\mathcal{M}'$ into a 2-counter machine $\mathcal{M}''$ using Gödel numbers [35]: indeed, the encoding preserves the trace set (un-)boundedness of $\mathcal{M}'$. □

4.2. Nondeterministic WSTS

Regarding nondeterministic WSTS with uniformly bounded branching, we reduce state boundedness for reset Petri nets, which is undecidable [25, Theorem 13], to trace boundedness for labeled reset Petri nets. From a reset Petri net we construct a labeled reset Petri net similar to that of Fig. 3, which hides the computation details thanks to a relabeling of the transitions. The new net consumes tokens using two concurrent, differently labeled transitions, so that the trace set can attest to state unboundedness.

Let us first recall that a marked *Petri net* is a tuple $\mathcal{N} = \langle P, \Theta, f, m_0\rangle$ where P and Θ are finite sets of places and transitions, f a flow function from $(P \times \Theta) \cup (\Theta \times P)$ to $\mathbb{N}$, and m_0 an initial marking in $\mathbb{N}^P$. The set of *markings* $\mathbb{N}^P$ is ordered component-wise by $m \le m'$ iff $\forall p \in P,\ m(p) \le m'(p)$, and has the zero marking $\mathbf{0}$ as least element, such that $\forall p \in P,\ \mathbf{0}(p) \stackrel{\text{def}}{=} 0$. A transition $t \in \Theta$ can be fired in a marking m if $f(p,t) \ge m(p)$ for all $p \in P$, and reaches a new marking m' defined by $m'(p) \stackrel{\text{def}}{=} m(p) - f(p,t) + f(t,p)$ for all $p \in P$.

A *labeled* Petri net (without ε labels) further associates a labeling letter-to-letter homomorphism $\sigma : \Theta \to \Sigma$, and can be seen as a finite branching WSTS $\langle \mathbb{N}^P, m_0, \Sigma, \to, \le\rangle$ where $m \xrightarrow{\sigma(t)} m'$ if the transition t can be fired in m and reaches m'. Determinism of such a system is decidable in ExpSpace [41]. An important class of deterministic Petri nets is defined by setting $\Sigma = \Theta$ and $\sigma = \mathrm{id}_\Theta$, thereby obtaining the so-called *free labeled* Petri nets.

A *reset* Petri net $\mathcal{N} = \langle P, \Theta, R, f, m_0\rangle$ is a Petri net $\langle P, \Theta, f, m_0\rangle$ with a set $R \subseteq P \times \Theta$ of reset arcs. The marking m' reached after a transition t from some marking m is now defined for all p in P by

$$m'(p) \stackrel{\text{def}}{=} \begin{cases} f(t,p) & \text{if } (p,t) \in R \\ m(p) - f(p,t) + f(t,p) & \text{otherwise.} \end{cases}$$

Proposition 15. *Trace boundedness is undecidable for labeled reset Petri nets.*

Proof. Let $\mathcal{N} = \langle P, \Theta, R, f, m_0\rangle$ be a reset Petri net. We construct a σ-labeled reset Petri net $\mathcal{N}'$ which is bounded if and only if $\mathcal{N}$ is state bounded, thereby reducing the undecidable problem of state boundedness in reset Petri nets [30].

We construct $\mathcal{N}'$ from $\mathcal{N}$ by adding two new places p_+ and p_-, two sets of new transitions t_p^c and t_p^d for each p in P, where each t_p^a for a in $\{c,d\}$ consumes one token from p_- and from p and puts one back in p_-, and one new transition t_- that takes one token from p_+ and puts it in p_-. All the transitions of $\mathcal{N}$ are modified to take one token from p_+ and put it back. Finally, we set $m_0(p_+) = 1$ and $m_0(p_-) = 0$ in the new initial marking. The labeling homomorphism σ from $\Theta \uplus \{t_-\} \uplus \{t_p^a \mid a \in \{c,d\}, p \in P\}$ to $\{a,b,c,d\}$ is defined by $\sigma(t) = a$ for all $t \in \Theta$, $\sigma(t_-) = b$, $\sigma(t_p^c) = c$ and $\sigma(t_p^d) = d$ for all p in P. See Fig. 6 for a pictorial representation of $\mathcal{N}'$. Its behavior is to simulate $\mathcal{N}$ while a token is in p_+ with a^* for trace, and to switch nondeterministically to a consuming behavior when transferring this token to p_- through t_-. Then, $\mathcal{N}'$ consumes tokens from the places of $\mathcal{N}$ and produces strings in $\{c,d\}^*$ through the t_p^c and t_p^d transitions.

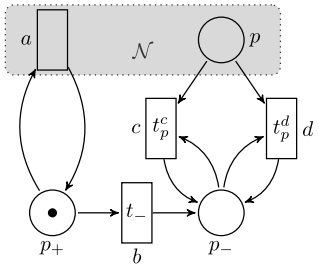

Fig. 6. The labeled reset Petri net $\mathcal{N}'$ of the proof of Proposition 15.

If $\mathcal{N}$ is not state bounded, then $\sum_{p\in P} m(p)$ for reachable markings m of $\mathcal{N}'$ is not bounded either. Thus an arbitrary number of t_p^c and t_p^d transitions can be fired, resulting in a trace set containing any string in $\{c,d\}^*$ as suffix for $\mathcal{N}'$, which entails that it is not trace bounded. Conversely, if $\mathcal{N}$ is trace bounded, then $\sum_{p\in P} m(p)$ is bounded by some constant n for all the reachable markings m of $\mathcal{N}'$, hence $T(\mathcal{N}')$ is included in the set of prefixes of $a^*b\{c,d\}^n$, a bounded language. □

4.3. Trace vs. post* flattability

The decidability of trace boundedness calls for the investigation of the decidability of less restrictive properties. Two natural candidates are post* flattability, which was proven undecidable for Minsky machines by Bardin et al. [4], and cover flattability, which is already known to be undecidable for cd-WSTS [2].

We show that post* flattability is still undecidable for cd-WSTS. To this end, we reduce again from state boundedness, this time in lossy channel systems [25], to post* flattability in an unlabeled *functional* lossy channel system, a deterministic variant introduced by Finkel and Goubault-Larrecq [14]. Somewhat analogously to Proposition 15, the idea is to consume the channel contents on one end while adding an unbounded sequence to its other end, so that the set of reachable configurations reveals state unboundedness.

A *lossy channel system* (LCS) is a WSTS $\mathcal{C} = \langle Q \times M^*, (q_0,\varepsilon), \{!,?\}\times M, \to, \preceq\rangle$ where Q is a finite set of states, $q_0 \in Q$ the initial state, M a finite set of messages, $(q,w) \preceq (q',w')$ if $q = q'$ and $w \preceq w'$—the scattered subword relation—and where the transition relation is defined from a finite relation $\delta \subseteq Q \times \{!,?\} \times M \times Q$ with

$$(q,w) \xrightarrow{!a} (q',w') \qquad \text{if } (q,!,a,q') \in \delta \text{ and } \exists w'' \in M^*,\ w'' \preceq w \text{ and } w' \preceq w''a$$

$$(q,w) \xrightarrow{?a} (q',w') \qquad \text{if } (q,?,a,q') \in \delta \text{ and } \exists w'' \in M^*,\ aw'' \preceq w \text{ and } w' \preceq w''.$$

One can easily extend this definition to accommodate for a finite set of channels and no-op transitions.

A *functional lossy channel system* [14] is defined in the same way except for the transition relation, which is now a partial function:

$$(q,w) \xrightarrow{!a} (q',wa) \qquad \text{if } (q,!,a,q') \in \delta$$

$$(q,uaw) \xrightarrow{?a} (q',w) \qquad \text{if } (q,?,a,q') \in \delta \text{ and } u \in (M\setminus\{a\})^*.$$

A functional LCS thus loses its channel contents lazily. There are some immediate relations between a LCS $\mathcal{C}$ and its corresponding functional LCS $\mathcal{C}'$, i.e. for the same Q, M, and δ: $\mathsf{Post}^*_{\mathcal{C}'}((q_0,\varepsilon)) \subseteq \mathsf{Post}^*_{\mathcal{C}}((q_0,\varepsilon))$, $\mathsf{Cover}_{\mathcal{C}'}((q_0,\varepsilon)) = \mathsf{Cover}_{\mathcal{C}}((q_0,\varepsilon))$, and $T(\mathcal{C}') = T(\mathcal{C})$.

Note that the following proposition is *not* a trivial consequence of the undecidability of cover-flattability in LCS [2], since in the case of functional LCS the Cover and Post* sets do not coincide.

Proposition 16. *Post* flattability is undecidable for functional lossy channel systems.*

Proof. Let us consider a LCS $\mathcal{C} = \langle Q \times M^*, (q_0,\varepsilon), \{!,?\}\times M, \to, \preceq\rangle$ and its associated functional system $\mathcal{C}'$. We construct a new functional LCS $\mathcal{C}''$ which is post* flattable if and only if $\mathcal{C}$ is state bounded, thereby reducing from the undecidable state boundedness problem for lossy channel systems [30]. Let us first remark that $\mathcal{C}$ is state bounded if and only if $\mathcal{C}'$ is state bounded, if and only if there is a maximal length n to the channel content w in any reachable configuration $(q,w) \in \mathsf{post}^*_{\mathcal{C}'}((q_0,\varepsilon))$.

We construct $\mathcal{C}''$ by adding two new states $q_?$ and $q_!$ to Q, two new messages c and d to M, and a set of new transitions to δ:

Table 1
Summary of complexity results for trace boundedness.

Petri nets	Affine counter systems	Functional LCS
ExpSpace-complete	Ack-complete	HAck-complete

$$\{(q, ?, a, q_!) \mid a \in M, q \in Q\}$$
$$\cup \{(q_!, !, a, q_?) \mid a \in \{c, d\}\}$$
$$\cup \{(q_?, ?, a, q_!) \mid a \in M\}\ .$$

If $\mathcal{C}'$ is state bounded, the writing transitions from $q_!$ can only be fired up to n times since they are interspersed with reading transitions from $q_?$, hence $\mathcal{C}''$ has its channel content lengths bounded by n. Therefore, $\mathcal{C}''$ is equivalent to a DFA with $(Q \uplus \{q_!, q_?\}) \times (M \uplus \{c, d\})^{\leq n}$ as state set and $\{!, ?\} \times (M \uplus \{c, d\})$ as alphabet. By removing all the loops via a depth-first traversal from the initial configuration (q_0, ε), we obtain a DFA $\mathcal{A}$ with a finite—and thus bounded—language, but with the same set of reachable states. Hence $\mathcal{C}''$ is post* flattable using $\mathcal{A}$.

Conversely, if $\mathcal{C}'$ is not state bounded, then an arbitrarily long channel content can be obtained in $\mathcal{C}''$, before performing a transition to $q_!$ and producing an arbitrarily long sequence in $\{c, d\}^*$ in the channel of $\mathcal{C}''$, witnessing an unbounded trace suffix. Observe that, due to the functional semantics, $\mathcal{C}''$ has no means to remove these symbols, thus it has to put them in the channel in the proper order, by firing the transitions from $q_!$ in the same order. Therefore no DFA with a bounded language can be synchronized with $\mathcal{C}''$ and still allow all these configurations to be reached: $\mathcal{C}''$ is not post* flattable. □

5. Complexity of trace boundedness

Well-structured transition systems are a highly abstract class of systems, for which no complexity upper bounds can be given in general. Nevertheless, it is possible to provide precise bounds for several concrete classes of WSTS, and even to employ generic proof techniques to this end. Table 1 sums up our complexity results, using the *fast-growing* complexity classes of [42].

Fast Growing Hierarchy. Our complexity bounds are often adequately expressed in terms of a family of fast growing functions, namely the generators $(F_\alpha)_\alpha$ of the *Fast Growing Hierarchy* [43], which form a hierarchy of ordinal-indexed functions $\mathbb{N} \to \mathbb{N}$. The first non-primitive-recursive function of the hierarchy is obtained for $\alpha = \omega$, $F_\omega(n) = F_{n+1}(n)$ being a variant of the Ackermann function, and eventually majorizes any primitive-recursive function. Similarly, the first non-multiply-recursive function is defined by $\alpha = \omega^\omega$ and eventually majorizes any multiply-recursive function.

Following [42], we define $\mathbf{F}_\alpha$ as the class of problems decidable using resources bounded by $O(F_\alpha(p(n)))$ for instance size n and some reasonable function p (formally, p in $\bigcup_{\beta<\alpha} \mathcal{F}_\beta$ using the *extended Grzegorczyk hierarchy* [43]). Since F_3 is already non-elementary, the traditional distinctions between space and time, or between deterministic computations and nondeterministic ones, are irrelevant. This gives rise to the *Ackermannian* complexity class Ack $\stackrel{\text{def}}{=} \mathbf{F}_\omega$ and the *hyper-Ackermannian* complexity class HAck $\stackrel{\text{def}}{=} \mathbf{F}_{\omega^\omega}$.

5.1. Lower bounds

Let us describe a generic recipe for establishing lower bounds: Given a system $\mathcal{S}$ that simulates a space-bounded Turing machine $\mathcal{M}$, hence with a finite number of different configurations n_c, assemble a new system $\mathcal{S}'$ that first nondeterministically computes some N up to n_c (this is also known as a "weak" computer for n_c), then simulates the runs of $\mathcal{S}$ but decreases some counter holding N at each transition. Thus $\mathcal{S}'$ terminates and has a bounded trace set, but still simulates $\mathcal{M}$. Now, add two loops on two different symbols a and b from the configurations that simulate the halting state of $\mathcal{M}$, and therefore obtain a system which is trace bounded if and only if $\mathcal{M}$ does not halt. Put differently, we reduce the control-state reachability problem in terminating systems to the trace boundedness problem.

We instantiate this recipe in the cases of Petri nets in Section 5.1.1, using Lipton's [44] results, for reset Petri nets (and thus affine counter systems) in Section 5.1.2 using Schnoebelen's [45] results, and for lossy channel systems in Section 5.1.3, using Chambart and Schnoebelen's [22] results. Although the complexity for Petri nets is quite significantly lower than for the other classes of systems, we also derive a non-primitive-recursive lower bound on the *size* of a bounded expression for a trace bounded Petri net (Section 5.1.4).

5.1.1. ExpSpace-*hardness for Petri nets*

Let us first observe that, since Karp and Miller-like [16] constructions always terminate in Petri nets, the search for an increasing fork is an algorithm (instead of a semi-algorithm). However, the complexity of this algorithm is not primitive-recursive [46].

Meanwhile, we extend the ExpSpace-hardness result of Lipton [44] for the Petri net coverability problem to the trace boundedness problem. As shown in [23] using an extension of the techniques of Rackoff [47], trace boundedness is in ExpSpace for Petri nets, thus trace boundedness is ExpSpace-complete for Petri nets.

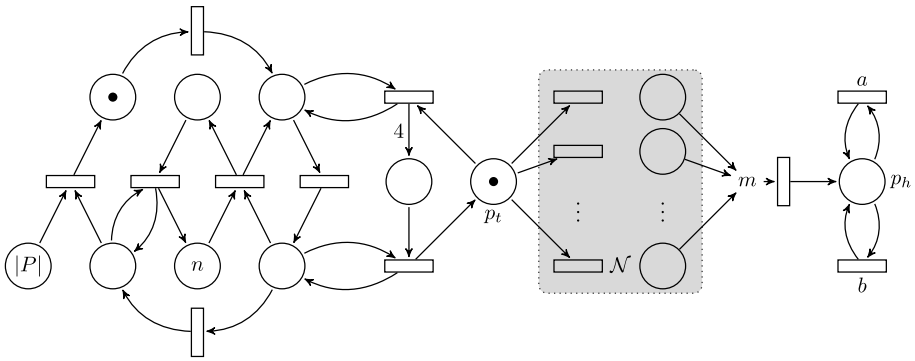

Fig. 7. The Petri net $\mathcal{N}'$ of the proof of Proposition 17.

Proposition 17. *Deciding the trace boundedness of a deterministic Petri net is* ExpSpace*-hard.*

Proof. The ExpSpace hardness of deciding whether a Petri net has a bounded trace set can be shown by adapting a well-known construction by Lipton [44]—see also the description given by Esparza [48]—for the ExpSpace-hardness of the coverability problem in Petri nets. We refer the reader to their construction of an $O(n^2)$-sized 2^{2^n}-bounded Petri net $\mathcal{N}$ that weakly simulates a 2^n-space bounded Turing machine $\mathcal{M}$, such that a marking greater than some marking m can be reached in $\mathcal{N}$ if and only if $\mathcal{M}$ halts.

We construct a new free labeled Petri net $\mathcal{N}'$ from $\mathcal{N} = \langle P, \Theta, f, m_0 \rangle$ and the marking m. Since the places in $\mathcal{N}$ are bounded by 2^{2^n}, only $n_c \stackrel{\text{def}}{=} 2^{2^{n|P|}}$ different configurations are reachable from m_0 in $\mathcal{N}$, therefore we can limit the length of all the computations in $\mathcal{N}$ to n_c and still obtain the same reachability set.

We initially plug a subnet that "weakly" computes some $N \leq n_c$ in a new place p_t (displayed in the left part of Fig. 7), in less than kn_c steps for some constant k. This subnet only uses a constant size and an initial submarking of size $O(|P| + n)$. We then simulate $\mathcal{N}$ but modify its transitions to consume one token from p_t each time. Finally, a new transition that consumes m from the subnet for $\mathcal{N}$ adds one token in another new place p_h that allows two new different transitions a and b to be fired at will; see Fig. 7.

A run of $\mathcal{N}'$ either reaches p_h and can then have any string in $\{a, b\}^*$ as a suffix, or is of length bounded by $(k+1)n_c$. Hence, $T(\mathcal{N}')$ is trace unbounded if and only if a run of $\mathcal{N}$ reaches some $m' \geq m$, if and only if the 2^n-space bounded Turing machine $\mathcal{M}$ halts, which proves the ExpSpace-hardness of deciding the trace boundedness of a Petri net. □

5.1.2. Ack*-hardness for affine counter systems*

Schnoebelen [45] shows that reset Petri nets (and thus affine counter systems) can simulate Minsky machines with counters bounded by $F_k(x)$ for some finite k and x. Thus we can encode a $F_\omega(n)$ space-bounded Turing machine using a $2^{F_\omega(n)}$-bounded Minsky machine. Since

$$2^{F_\omega(n)} = 2^{F_{n+1}(n)} \leq F_2(F_{n+1}(n)) \leq F^2_{n+2}(n) \leq F_{n+3}(n+1) \ ,$$

we can simulate this Minsky machine with a polynomial-sized reset Petri net, and we get:

Proposition 18. *Trace boundedness of reset Petri nets is not primitive-recursive, more precisely it is hard for* Ack*.*

Proof sketch. The construction is almost exactly the same as for the proof of Schnoebelen's [45] Theorem 7.1 of hardness of termination. One simply has to replace extended instructions using reset transitions as explained in Schnoebelen's [45] Section 6, and to replace the single outgoing transition on ℓ_ω by two different transitions, therefore yielding an unbounded trace set. □

5.1.3. HAck*-hardness for lossy channel systems*

Chambart and Schnoebelen [22] show that LCS can weakly compute any multiply-recursive function, and manage to simulate perfect channel systems (i.e. Turing machines) of size bounded by such functions, thereby obtaining a non-multiply-recursive lower bound for LCS reachability. We prove that the same bound holds for trace boundedness.

Proposition 19. *Trace boundedness of functional lossy channel systems is not multiply-recursive, more precisely it is hard for* HAck*.*

Proof. Chambart and Schnoebelen [22] show that it is possible to perfectly simulate a Turing machine $\mathcal{M}$ with input x and $k = |x|$ that works in space bounded by $F_{\omega^\omega}(k) = F_{\omega^{k+1}}(k)$, with an LCS $\mathcal{C}_\mathcal{M}$ of size polynomial in k and $|\mathcal{M}|$, such that a state q_f of $\mathcal{C}_\mathcal{M}$ is reachable if and only if $\mathcal{M}$ halts. Furthermore, the number of distinct configurations $n_c = |Q| \cdot |M|^{F_{\omega^{k+1}}(k)}$

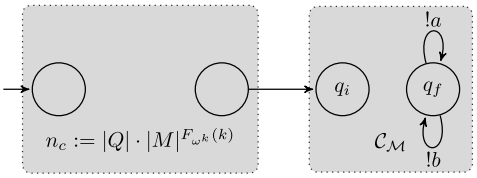

Fig. 8. The lossy channel system $\mathcal{C}'_{\mathcal{M}}$ for the proof of Proposition 19.

of $\mathcal{C}_{\mathcal{M}}$ can also be weakly computed in unary with an LCS of polynomial size, Q being the set of states of $\mathcal{C}_{\mathcal{M}}$ and M its message alphabet.

Combining those two systems, we construct $\mathcal{C}'_{\mathcal{M}}$ (see Fig. 8) that

1. first "weakly" computes some $N \leq n_c$ (in a separate channel with a unary alphabet), and then
2. executes $\mathcal{C}_{\mathcal{M}}$ while decrementing N at each transition step,
3. is able to loop on two added transitions $q_f \xrightarrow{!a} q_f$ and $q_f \xrightarrow{!b} q_f$, which do not decrement N, giving rise to an unbounded trace.

In a nutshell, all the runs of $\mathcal{C}'_{\mathcal{M}}$ that do not visit q_f are terminating, being of length bounded by n_c. Consequently, $\mathcal{C}'_{\mathcal{M}}$ is unbounded if and only if q_f is reachable, if and only if it was also reachable in $\mathcal{C}_{\mathcal{M}}$, if and only if $\mathcal{M}$ halts.

We conclude the proof by remarking that both the weak computation of n_c and the perfect simulation of $\mathcal{M}$ keep working with the functional lossy semantics. □

5.1.4. Non primitive-recursive size of a bounded expression for Petri nets

We derive a non-primitive-recursive lower bound on the computation of the words $w_1, \ldots, w_n$, already in the case of Petri nets. Indeed, the size of a covering tree can be non-primitive-recursive compared to the size of the Petri net [46, who attribute the idea to Hack]. Using the same insight, we demonstrate that the words $w_1, \ldots, w_n$ themselves can be of non-primitive-recursive size. This complexity is thus inherent to the computation of the w_i's.

Proposition 20. *There exists a free labeled Petri net $\mathcal{N}$ with a bounded trace set $T(\mathcal{N})$ but such that for any words $w_1, \ldots, w_n$, if $T(\mathcal{N}) \subseteq w_1^* \cdots w_n^*$, then the size $\sum_{i=1}^n |w_i|$ is not primitive-recursive in the size of $\mathcal{N}$.*

Proof. We consider for this proof a Petri net that "weakly" computes a non-primitive-recursive function $A : \mathbb{N} \to \mathbb{N}$. The particular example displayed in Fig. 9 is taken from a survey by Jantzen [49], where A is defined for all m and n by

$$A(n) \stackrel{\text{def}}{=} A'_n(2) \qquad A'_0(n) \stackrel{\text{def}}{=} 2n+1$$
$$A'_{m+1}(0) \stackrel{\text{def}}{=} 1 \qquad A'_{m+1}(n+1) \stackrel{\text{def}}{=} A'_m(A'_{m+1}(n)) \,.$$

The marked Petri net $\mathcal{N}$ for $A(n)$ is of linear size in n and its trace set L is finite, and therefore bounded, but contains words of non-primitive-recursive length compared to n.

Although it might seem intuitively clear that we need a collection of words $w_1, \ldots, w_n$ of non-primitive-recursive size in order to capture this trace set, the proof is slightly more involved. Observe for instance that the finite trace set $\{a^p\}$ where p is an arbitrary number is included in the bounded expression a^* of size $|a| = 1$. Thus there is no general upper bound to the ratio between the size $\sum_{w \in L} |w|$ of a finite trace set L and the size of the minimal collection of words that proves that L is bounded.

Let us consider the maximal run in the Petri net for $A(n)$. We focus on the two black transitions labeled a and b in Fig. 9, and more precisely on the suffix of the run where we compute $A'_n(2) = A'_1(p)$ with

$$p = A'_2(A'_3(\ldots(A'_n(1) - 1)\ldots) - 1) \,.$$

This computation takes place in the subnet for A'_0 and A'_1 solely, and this suffix is of form $v = ab^{k_0}ab^{k_1}\cdots ab^{k_p}$ with $k_0 = 1$, $k_{i+1} = 2k_i + 1$, and $k_p = A'_1(p) = A'_n(2)$. By Lemma 13 any bounded expression such that $v \in w_1^* \cdots w_n^*$ has size $\sum_{i=1}^n |w_i| > p$.

We conclude by noting (1) that p is already the image of n by a non-primitive-recursive function, and (2) that v is the suffix of the projection u of a word in $T(\mathcal{N})$ on the alphabet $\{a, b\}$: hence, if a bounded expression of primitive-recursive size with $T(\mathcal{N}) \subseteq w_1^* \cdots w_n^*$ existed, then the projections w'_i of the w_i on $\{a, b\}$ would be such that $|w'_i| \leq |w_i|$ and $u \in w'^*_1 \cdots w'^*_n$, and would yield an expression of primitive-recursive size for v. □

In the case of Petri nets we are in a situation comparable to that of context-free languages: trace boundedness is decidable with a sensibly smaller complexity than the complexity of the size of the corresponding bounded expression (see Gawrychowski et al. [50] for a PTIME algorithm for deciding trace boundedness of a context-free grammar, and Habermehl and Mayr [51] for an example of an expression exponentially larger than the grammar).

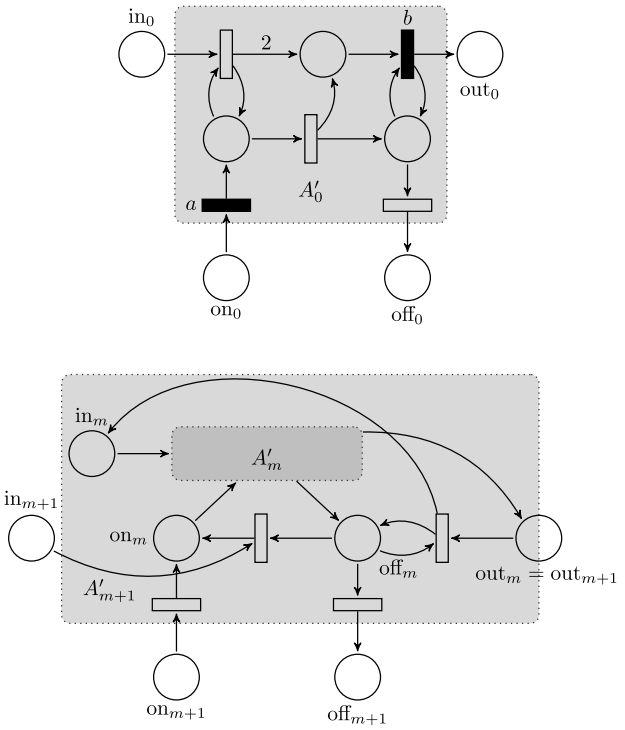

Fig. 9. A Petri net that "weakly" computes A'_{m+1} [49].

5.2. Upper bounds

We provide another recipe, this time for proving upper bounds for trace-boundedness in cd-WSTS, relying on existing *length function theorems* on wqos, which prove upper bounds on the length of *controlled bad sequences*.

Controlled good and bad sequences. Let $(S, \leq)$ be a quasi order. A sequence $s_0 \cdots s_\ell$ in S^* is *r-good* if there exist $0 \leq i_0 < i_1 < \cdots < i_r \leq \ell$ with $s_{i_j} \leq s_{i_{j+1}}$ for all $0 \leq j < r$, and is *r-bad* otherwise. In the case $r = 1$, we say more simply that the sequence is *good* (resp. *bad*). The wqo condition thus ensures that any infinite sequence is good.

Given a *norm* function $\|.\| S \to \mathbb{N}$ with $S_{\leq n} \stackrel{\text{def}}{=} \{s \in S \mid \|s\| \leq n\}$ finite for every n, a *control* function $g\colon \mathbb{N} \to \mathbb{N}$, g monotone s.t. $g(x) > x$ for all x, and an *initial norm* n in $\mathbb{N}$, a sequence $s_0 \cdots s_\ell$ is *controlled* by $(\|.\|, g, n)$ if, for all i, $\|s_i\| \leq g^i(n)$ the ith iteration of g; in particular, $\|s_0\| \leq n$ initially.

A cd-WSTS $\langle S, s_0, \Sigma, \to, \leq \rangle$ is *(strongly) controlled* by $(\|.\|, g, n)$ if

1. $\|s_0\| \leq n$,
2. for any single step $s \xrightarrow{a} s'$, $\|s'\| \leq g(\|s\|)$, and
3. for any accelerated step $s \xrightarrow{u^\omega} s'$, $\|s'\| \leq g^{|u|}(\|s\|)$.

Using these notions, and by a careful analysis of the proof of Proposition 5, we exhibit in Section 5.2.1 a witness of trace unboundedness under the form of a good $(\|.\|, g^2, n)$-controlled sequence $s_0 \cdots s_\ell$ of S^* in a $(\|.\|, g, n)$-controlled WSTS. There is therefore a longest bad prefix to this witness, which is still controlled.

The particular way of generating this sequence yields an algorithm, since as a consequence of the wqo, the depth of exploration in the search for this witness of trace unboundedness is finite, and we can therefore replace the two semi-algorithms of Section 3 by a single algorithm that performs an exhaustive search up to this depth. Furthermore, we can apply *length function theorems* to obtain upper bounds on the maximal length of bad controlled sequences, and thus on this depth; this is how the upper bounds of Table 1 are obtained (see Section 5.2.2 and Section 5.2.3).

5.2.1. Extracting a controlled good sequence

Let us assume we are given a trace unbounded $(\|.\|, g, n)$-controlled cd-WSTS $\mathcal{S}$, and let us consider the three infinite sequences defined in the proof of Lemma 11, namely $(v_i, u_i)_{i>0}$ of pairs of words in $\Sigma^* \times \Sigma^+$, $(L_i)_{i\geq 0}$ of unbounded languages, and $(s_i)_{i\geq 0}$ of states starting with the initial state s_0. By construction, $(s_i)_{i\geq 0}$ is good; however, this sequence is not controlled by a "reasonable" function in terms of g, because we use the wqo argument at each step (when we employ Lemma 10 to construct s_{i+1} from s_i), hence the motivation for refining this first sequence. A solution is to also consider

some of the intermediate configurations along the transition sequence $v_{i+1}u_{i+1}$ starting in s_i, so that the index of each state in the new sequence better reflects how the state was obtained.

Lemma 21. *Let $\mathcal{S} = \langle S, s_0, \Sigma, \to, \leq\rangle$ be a $(\|.\|, g, n)$-controlled cd-WSTS. Then we can construct a specific $(\|.\|, g^2, n)$-controlled sequence which is good if and only if $\mathcal{S}$ is trace unbounded.*

Proof. As in the proof of Lemma 11, we construct inductively on i the following three infinite sequences $(v_i, u_i)_{i>0}$, $(L_i)_{i\geq 0}$ starting with $L_0 \stackrel{\text{def}}{=} T(\mathcal{S})$, and $(s_i)_{i\geq 0}$ starting with the initial state s_0 of $\mathcal{S}$, such that

- v_{i+1}, u_{i+1} are chosen using Lemma 10 such that
  1. $v_{i+1}u_{i+1}^{\omega}$ is in $T_{\text{acc}}(\mathcal{S}(s_i))$,
  2. $v_{i+1}u_{i+1}$ is in $\mathsf{Pref}(L_i)$,
  3. $|v_{i+1}| \geq |u_i|$ if $i > 0$ (and thus $|v_{i+1}u_{i+1}| \geq |u_i|$ as in the proof of Lemma 11),
  4. $\overline{u_{i+1}}(v_{i+1}^{-1}L_i)$ is unbounded, and
  5. there do not exist two successive strict prefixes p, p' of $v_{i+1}u_{i+1}$ such that $|p| \geq |u_i|$ and $s_i \xrightarrow{p} s_i' \xrightarrow{p'} s_i''$ with $s_i' \leq s_i''$, i.e. $v_{i+1}u_{i+1}$ is the shortest choice for Lemma 10 and (1–4) above;
- $s_i \xrightarrow{v_{i+1}u_{i+1}^{\omega}} s_{i+1}$;
- $L_{i+1} \stackrel{\text{def}}{=} \overline{u_{i+1}}(v_{i+1}^{-1}L_i)$.

We define another sequence of states $(s_{i,j})_{i\geq 0, j\in J_i}$ by $s_i \xrightarrow{p_{i,j}} s_{i,j}$ with $p_{i,j}$ the prefix of length j of $v_{i+1}u_{i+1}$, where

$$J_0 \stackrel{\text{def}}{=} \{0, \ldots, |v_1u_1| - 1\} \text{ and}$$
$$J_i \stackrel{\text{def}}{=} \{|u_i|, \ldots, |v_{i+1}u_{i+1}| - 1\} \text{ for } i > 0.$$

Because $|u_i| > 0$ for each $i > 0$, none of the $(s_i)_{i>0}$ appears in the sequence $(s_{i,j})_{i\geq 0, j\in J_i}$. Note that condition (5) on the choice of $v_{i+1}u_{i+1}$ ensures that, for each $i \geq 0$, each factor $(s_{i,j})_{j\in J_i}$ is a bad sequence.

This infinite sequence of states $(s_{i,j})_{i\geq 0, j\in J_i}$ can be constructed whenever we are given a trace unbounded cd-WSTS, and is necessarily *good* due to the wqo. Our aim will be later to bound the length of its longest bad prefix. In order to do so, we need to control this sequence:

Claim 21.1. *The sequence $(s_{i,j})_{i\geq 0, j\in J_i}$ is controlled by $(\|.\|, g^2, n)$.*

Proof. Since $\mathcal{S}$ is $(\|.\|, g, n)$-controlled, we can control the accelerated transition sequence that led to a given $s_{i,j}$: first reach s_i, and then apply j single step transitions. Formally, put for all $i \geq 0$

$$k_0 \stackrel{\text{def}}{=} 0, \qquad k_{i+1} \stackrel{\text{def}}{=} k_i + |v_{i+1}| + |u_{i+1}|\,,$$

where $|v_{i+1}|$ accounts for the single steps and $|u_{i+1}|$ for the accelerated step in $s_i \xrightarrow{v_{i+1}u_{i+1}^{\omega}} s_{i+1}$; then we have for all $i \geq 0$ and $j \in J_i$

$$\|s_{i,j}\| \leq g^{k_i+j}(n)\,.$$

We need to relate this norm with the index of each $s_{i,j}$ in the $(s_{i,j})_{i\geq 0, j\in J_i}$ sequence. We define accordingly for all $i \geq 0$ and $j \in J_i$

$$\ell_{0,\min J_0} \stackrel{\text{def}}{=} 0, \qquad \ell_{i,j+1} \stackrel{\text{def}}{=} \ell_{i,j} + 1, \qquad \ell_{i+1,\min J_{i+1}} \stackrel{\text{def}}{=} \ell_{i,\min J_i} + |J_i|\,.$$

In order to prove our claim, namely that

$$\|s_{i,j}\| \leq (g^2)^{\ell_{i,j}}(n)\,,$$

we show by induction on (i, j) ordered lexicographically that

$$k_i + j \leq 2 \cdot \ell_{i,j}\,.$$

The base case for $i = 0$ and $j = \min J_0 = 0$ is immediate, since $k_i + j = 0 = 2 \cdot \ell_{0,0}$. For the induction step on j, $k_i + j + 1 \leq 2 \cdot \ell_{i,j} + 1 \leq 2 \cdot \ell_{i,j+1}$, and for the induction step on i,

$$\begin{aligned} k_{i+1} + \min J_{i+1} &= k_{i+1} + |u_{i+1}| && \text{(by def. of } J_{i+1}) \\ &= k_i + 2|u_{i+1}| + |v_{i+1}| && \text{(by def. of } k_{i+1}) \\ &= k_i + |u_i| + 2|u_{i+1}| + |v_{i+1}| - |u_i| \end{aligned}$$

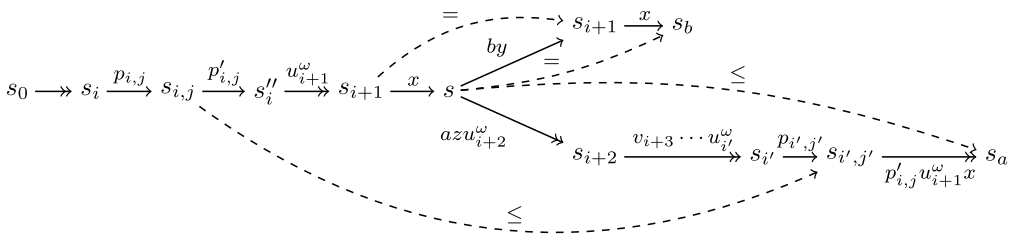

Fig. 10. The construction of an increasing fork in the proof of Claim 21.2.

$$
\begin{aligned}
&= k_i + \min J_i + 2|u_{i+1}| + |v_{i+1}| - |u_i| && \text{(by def. of } J_i\text{)}\\
&\leq 2 \cdot \ell_{i,\min J_i} + 2|u_{i+1}| + |v_{i+1}| - |u_i| && \text{(by ind. hyp.)}\\
&\leq 2 \cdot \ell_{i,\min J_i} + 2|u_{i+1}| + 2|v_{i+1}| - 2|u_i| && \text{(since } |v_{i+1}| \geq |u_i|\text{)}\\
&= 2 \cdot \ell_{i+1,\min J_{i+1}} \,. && \text{(by def. of } \ell_{i+1,\min J_{i+1}}\text{)}
\end{aligned}
$$

Thus by monotonicity of g,

$$\|s_{i,j}\| \leq g^{k_i+j}(n) \leq g^{2\cdot\ell_{i,j}}(n) \,. \qquad \square$$

We also need to show that such a good sequence is a *witness* for trace unboundedness, which we obtain thanks to Lemma 7 and the following claim:

Claim 21.2. *If the sequence $(s_{i,j})_{i\geq 0, j\in J_i}$ is good, then $\mathcal{S}$ has an increasing fork.*

Proof. Let $s_{i,j}$ and $s_{i',j'}$ be two elements of the sequence witnessing goodness, such that $s_{i,j}$ occurs before $s_{i',j'}$ and $s_{i,j} \leq s_{i',j'}$. Due to the constraints put on the choices of v_{i+1} and u_{i+1} for each i, we know that $i < i'$. Similarly to the proof of Lemma 11, there exists a longest common prefix x in Σ^* and two symbols $a \neq b$ in Σ such that $v_{i+2}u_{i+2} = xaz$ and $u_{i+1} = xby$ for some y and z in Σ^*. Let us further call $p'_{i,j}$ the suffix of $v_{i+1}u_{i+1}$ such that $v_{i+1}u_{i+1} = p_{i,j}p'_{i,j}$, hence we get a fork by selecting s, s_a, and s_b with

$$s_{i,j} \xrightarrow{p'_{i,j}u^\omega_{i+1}x} s \qquad s \xrightarrow{azu^\omega_{i+2}\cdots v_{i'}u^\omega_{i'}p_{i',j'}} s_{i',j'} \xrightarrow{p'_{i,j}u^\omega_{i+1}x} s_a \qquad s \xrightarrow{byx} s_b \,.$$

Note that because $|x| < |u_{i+1}|$ and $i < i'$, $s_{i',j'}$ is necessarily met after s and the construction is correct. See also Fig. 10. $\square$

This concludes the proof of the lemma: $\mathcal{S}$ is trace unbounded if and only if $(s_{i,j})_{i\geq 0, j\in J_i}$ is good. $\square$

In the following, we essentially bound the complexity of trace boundedness using bounds on the length of the $(s_{i,j})_{i\geq 0, j\in J_i}$ sequence. This is correct modulo a few assumptions on the concrete systems we consider, and because the fast growing upper bounds we obtain dwarfen any additional complexity sources. For instance, a natural assumption would be for the size of representation of an element s of S to be less than $\|s\|$, but actually any primitive-recursive function of $\|s\|$ would still yield the same upper bounds!

5.2.2. $\mathbf{F}_\omega$ *upper bound for affine counter systems*

We match the Ack lower bound of Proposition 18 for affine counter systems, thus establishing that trace boundedness is Ack-complete. We employ the machinery of Claims 21.1 and 21.2, and proceed by showing that

1. complete affine counter systems are controlled, and that
2. one can provide an upper bound on the length of bad sequences in $(\mathbb{N} \uplus \{\omega\})^k$.

Controlling complete affine counter systems. Recall that an *affine counter system* (ACS) $\langle L, \boldsymbol{x}_0\rangle$ is a finite set L of affine transition functions of form $f(\boldsymbol{x}) = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{b}$, with $\boldsymbol{A}$ a matrix in $\mathbb{N}^{k\times k}$ and $\boldsymbol{b}$ a vector in $\mathbb{Z}^k$, along with an initial configuration $\boldsymbol{x}_0$ in $\mathbb{N}^k$. A transition f is firable in configuration $\boldsymbol{x}$ of $\mathbb{N}^k$ if $f(\boldsymbol{x}) \geq \boldsymbol{0}$, and leads to a new configuration $f(\boldsymbol{x})$.

Define the *norm* $\|\boldsymbol{x}\|$ of a configuration in $(\mathbb{N} \uplus \{\omega\})^k$ as the infinity norm among finite values $\|\boldsymbol{x}\| \stackrel{\text{def}}{=} \max(\{0\} \cup \{\boldsymbol{x}[j] \neq \omega \mid 1 \leq j \leq k\})$. Also set m_1 as the maximal coefficient

$$m_1 \stackrel{\text{def}}{=} \max_{f(\boldsymbol{x})=\boldsymbol{A}\boldsymbol{x}+\boldsymbol{b}\in L, 1\leq i,j\leq k} \boldsymbol{A}[i,j]$$

and m_2 as the maximal constant

$$m_2 \stackrel{\text{def}}{=} \max_{f(\boldsymbol{x})=\boldsymbol{A}\boldsymbol{x}+\boldsymbol{b}\in L, 1\le i\le k} \boldsymbol{b}[i]\,.$$

In case of a single step transition using some function f in L, one has

$$\|f(\boldsymbol{x})\| \le k \cdot m_1 \cdot \|x\| + m_2\,,$$

while in case of an accelerated transition sequence, one has the following:

Claim 22.1. *Let $u = f_n \circ \cdots \circ f_1$ be a transition sequence in L^+ with $u(\boldsymbol{x}) \ge \boldsymbol{x}$. Then $\|(u^\omega(\boldsymbol{x}))\| \le (k \cdot m_1)^{n\cdot k} \cdot (\|\boldsymbol{x}\| + n \cdot k \cdot m_2)$.*

Proof. We first proceed by proving that k iterations of u are enough in order to compute the finite values in $u^\omega(\boldsymbol{x})$.

Let us set $u(\boldsymbol{x}) = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{b}$ and $\boldsymbol{d}_n \stackrel{\text{def}}{=} u^{n+1}(\boldsymbol{x}) - u^n(\boldsymbol{x})$ for all n. Since $u(\boldsymbol{x}) \ge \boldsymbol{x}$, for any coordinate $1 \le j \le k$, the limit $\lim_{n\to\omega} u^n(\boldsymbol{x})[j]$ exists, and is finite if and only if $\boldsymbol{x}[j] < \omega$ and there exists m such that for all $n \ge m$, $\boldsymbol{d}_n[j] = 0$. As $\boldsymbol{d}_{n+1} = u^{n+2}(\boldsymbol{x}) - u^{n+1}(\boldsymbol{x}) = \boldsymbol{A} \cdot u^{n+1}(\boldsymbol{x}) + \boldsymbol{b} - (\boldsymbol{A} \cdot u^n(\boldsymbol{x}) + \boldsymbol{b}) = \boldsymbol{A} \cdot (u^{n+1}(\boldsymbol{x}) - u^n(\boldsymbol{x})) = \boldsymbol{A} \cdot \boldsymbol{d}_n$, we have $\boldsymbol{d}_n = \boldsymbol{A}^n \cdot \boldsymbol{d}_0$.

If we consider $\boldsymbol{A}$ as the adjacency matrix of a weighted graph with k vertices, its $\boldsymbol{A}^n[i, j]$ entry is the sum of the weights of all the paths $\psi = \psi_0\,\psi_1 \cdots \psi_n$ of length n through the matrix, which start from $\psi_0 = i$ and end in $\psi_n = j$, i.e.

$$\boldsymbol{A}^n[i, j] = \sum_{\psi\in\{i\}\times[1,k]^{n-1}\times\{j\}} \prod_{\ell\in[0,n-1]} \boldsymbol{A}[\psi_\ell, \psi_{\ell+1}]$$

$$\boldsymbol{d}_n[j] = \sum_{\psi\in\times[1,k]^n\times\{j\}} \left(\boldsymbol{d}_0[\psi_0] \cdot \prod_{\ell\in[0,n-1]} \boldsymbol{A}[\psi_\ell, \psi_{\ell+1}] \right).$$

Since $u(\boldsymbol{x}) \ge \boldsymbol{x}$ and $\boldsymbol{A}$ contains non-negative integers from $\mathbb{N}$, $\boldsymbol{d}_n[j] = \boldsymbol{0}$ iff each of the above products is null, iff there is no path of length n in the graph of $\boldsymbol{A}$ starting from a non-null $\boldsymbol{d}_0[i]$. Therefore, if there exists $n > k$ such that $\boldsymbol{d}_n[j] > 0$, then there is a path with a loop of positive weight in the graph. In such a case there are infinitely many m such that $\boldsymbol{d}_m[j] > 0$. A contrario, if there exists m such that for all $n \ge m$, $\boldsymbol{d}_n[j] = 0$, then $m = k$ is enough: if $u^\omega(\boldsymbol{x})[j] \in \mathbb{N}$, then $u^\omega(\boldsymbol{x})[j] = u^k(\boldsymbol{x})[j]$.

Let us now derive the desired upper bound on the norm of $u^\omega(\boldsymbol{x})$: either $u^\omega(\boldsymbol{x})[j] = \omega$ and the jth coordinate does not contribute to $\|u^\omega(\boldsymbol{x})\|$, or $u^\omega(\boldsymbol{x})[j] \in \mathbb{N}$ and $u^\omega(\boldsymbol{x})[j] = u^k(\boldsymbol{x})[j]$. Let $f_i(\boldsymbol{x}) \stackrel{\text{def}}{=} \boldsymbol{A}_i \cdot \boldsymbol{x} + \boldsymbol{b}_i$; we have

$$\boldsymbol{A} = \prod_{i=n}^{1} \boldsymbol{A}_i \qquad \boldsymbol{b} = \sum_{j=1}^{n} \left(\prod_{i=n}^{j+1} \boldsymbol{A}_i \right) \cdot \boldsymbol{b}_j$$

$$\begin{aligned} u^k(\boldsymbol{x}) &= \boldsymbol{A}^k \cdot \boldsymbol{x} + \sum_{\ell=0}^{k-1} \boldsymbol{A}^\ell \cdot \boldsymbol{b} \\ &= \left(\prod_{i=n}^{1} \boldsymbol{A}_i \right)^k \cdot \boldsymbol{x} + \sum_{\ell=0}^{k-1} \sum_{j=1}^{n} \left(\prod_{i=n}^{1} \boldsymbol{A}_i \right)^\ell \cdot \left(\prod_{i=n}^{j+1} \boldsymbol{A}_i \right) \cdot \boldsymbol{b}_j \end{aligned}$$

thus

$$\begin{aligned} \|u^\omega(\boldsymbol{x})[j]\| &\le \|\boldsymbol{A}^k \cdot \boldsymbol{x}\| + \sum_{j=0}^{k-1} \|\boldsymbol{A}^j \cdot \boldsymbol{b}\| \\ &\le (k \cdot m_1)^{n\cdot k} \cdot \|\boldsymbol{x}\| + n \cdot k \cdot (k \cdot m_1)^{n\cdot k} \cdot m_2 \\ &= (k \cdot m_1)^{n\cdot k} \cdot (\|\boldsymbol{x}\| + n \cdot k \cdot m_2) \qquad \square \end{aligned}$$

Length function theorem. It remains to apply the bounds of Figueira et al. [52] on the length of controlled r-bad sequences over $\mathbb{N}^k$:

Proposition 22. *Trace boundedness for affine counter systems is in* Ack.

Proof. Define the projections p_1 and p_2 from $(\mathbb{N} \uplus \{\omega\})$ to $\mathbb{N}$ and $\{1, \omega\}$ respectively by

$$p_1(\omega) \stackrel{\text{def}}{=} 0 \qquad p_1(n) \stackrel{\text{def}}{=} n \qquad p_2(\omega) \stackrel{\text{def}}{=} \omega \qquad p_2(n) \stackrel{\text{def}}{=} 1$$

for $n < \omega$, and their natural extensions from $(\mathbb{N} \uplus \{\omega\})^k$ to $\mathbb{N}^k$ and $\{1, \omega\}^k$.

Consider the projection $(\boldsymbol{x}_{i,j})_{i\geq 0, j\in J_i} = \big(p_1(s_{i,j})\big)_{i\geq 0, j\in J_i}$ on $\mathbb{N}^k$ of the sequence defined in Section 5.2.1. This sequence is $(\|.\|, g, \|\boldsymbol{x}_0\|)$-controlled if $(s_{i,j})_{i\geq 0, j\in J_i}$ is $(\|.\|, g, \|\boldsymbol{x}_0\|)$-controlled, and is r-good for any finite r whenever the trace set of the affine counter system is unbounded.

Conversely, if the sequence $(\boldsymbol{x}_{i,j})_{i\geq 0, j\in J_i}$ is 2^k-good for the product ordering $\leq$ on $\mathbb{N}^k$, then the system has an increasing fork. Indeed, let $r = 2^k$; by definition of a r-good sequence, we can extract an increasing chain $\boldsymbol{x}_{k_0} \leq \boldsymbol{x}_{k_1} \leq \cdots \leq \boldsymbol{x}_{k_r}$ from the sequence $(\boldsymbol{x}_{i,j})_{i\geq 0, j\in J_i}$. Since $r = 2^k$, there exist $k_i < k_j$ such that $p_2(s_{k_i}) = p_2(s_{k_j})$, and therefore $s_{k_i} \leq s_{k_j}$ and we can apply Claim 21.2 to construct an increasing fork.

By Claim 22.1, the sequence $(\boldsymbol{x}_{i,j})_{i\geq 0, j\in J_i}$ is $(\|.\|, g, \|\boldsymbol{x}_0\|)$-controlled by a primitive-recursive function g (that depends on the size $\|L\|$ of the affine counter system $\langle L, \boldsymbol{x}_0\rangle$ at hand), hence it is of length bounded by $F_\omega(p(k, \|L\|, \|\boldsymbol{x}_0\|))$ for some fixed primitive-recursive function p [52]. □

5.2.3. $\mathbf{F}_{\omega^\omega}$ *upper bound for lossy channel systems*

Proposition 19 established a HAcк lower bound for the trace boundedness problem in lossy channel systems. We match this lower bound, thus establishing that trace boundedness is HAcк-complete. As in Section 5.2.2, we need two results in order to instantiate our recipe for upper bounds: a control on complete functional LCS, and a miniaturization for their sequences of states.

Controlling complete functional LCS. According to Abdulla et al. [27], LCS queue contents on an alphabet M can be represented by *simple regular expressions* (SRE) over M, which are finite unions of *products* over M. Products, endowed with the language inclusion ordering, suffice for the completion of functional LCS [14, Section 5], and thus for the representation of the effect of accelerated sequences in functional LCS.

Products can be seen as finite sequences over a finite alphabet

$$\Pi_M = \{(a+\varepsilon) \mid a \in M\} \cup \{A^* \mid A \subseteq M\}$$

with $|\Pi_M| = 2^{|M|} + |M|$, with associated languages $L(a+\varepsilon) \stackrel{\text{def}}{=} \{a, \varepsilon\}$ and $L(A^*) \stackrel{\text{def}}{=} A^*$. We consider the scattered subword ordering $\preceq$ on Π_M^*, defined as usual by $a_1 \cdots a_m \preceq b_1 \cdots b_n$ if there exists a monotone injection $f : \{1, \ldots, m\} \to \{1, \ldots, n\}$ such that, for all $1 \leq i \leq n$, $a_i = b_{f(i)}$. The scattered subword ordering is compatible with language inclusion, thus we can consider the subword ordering instead of language inclusion in our completed functional LCS[1]:

Claim 24.1. *For all products π, π' in Π_M^*, $\pi \preceq \pi'$ implies $L(\pi) \subseteq L(\pi')$.*

Let us fix for the remainder of this section an arbitrary complete functional LCS $\mathcal{C} = \langle Q \times (\Pi_M)^*, (q_0, \varepsilon), \{!, ?\} \times M, \to, \leq\rangle$, where $\leq$ is defined on configurations in $Q \times (\Pi_M)^*$ by $(q, \pi) \leq (q', \pi')$ if $q = q'$ and $L(\pi) \subseteq L(\pi')$.

Claim 24.2. *Functional LCS are controlled by $(\|.\|, g, 0)$ with $\|q, \pi\| \stackrel{\text{def}}{=} |\pi|$ and $g(x) \stackrel{\text{def}}{=} 2^{x+2} + x$.*

Proof. The claim follows from the results of Abdulla et al. [27] on SREs. Let the current configuration be $s \stackrel{\text{def}}{=} (q, \pi)$.

In the case of a single transition step $s \xrightarrow{a} s'$, a product grows by at most one atomic expression $(a+\varepsilon)$ [27, Lemma 6.1].

In the case of an accelerated transition step $s \xrightarrow{u^\omega} s'$ on a sequence u, since $s \xrightarrow{u} s''$ with $s \leq s''$, we are in one of the first three subcases of the proof of Lemma 6.4 of Abdulla et al. [27]: the first two subcases yield the addition of an atomic expression A^*, while the third subcase adds at most $|u|^{|\pi|+2}$ atomic expressions of form $(a+\varepsilon)$. □

Length function theorem. Schmitz and Schnoebelen [53] give an upper bound on the least N such that any $(\|.\|, g, n)$-controlled sequence σ with $|\sigma| = N$ of elements in $(\Sigma^*, \preceq)$ is r-good.

Fact 23. *(See Schmitz and Schnoebelen, 2011, [53].) Let g be a primitive-recursive unary function and n in $\mathbb{N}$. Then, if σ is a $(\|.\|, g, n)$-controlled r-bad sequence of $(\Sigma^*, \preceq)$, then $|\sigma| \leq F^r_{\omega^{|\Sigma|-1}}(p(n))$ for some primitive-recursive p.*

Proposition 24. *Trace boundedness for functional LCS is in* HAcк.

Proof. We consider the sequence of products $(\pi_{i,j})_{i\geq 0, j\in J_i}$ extracted from the sequence of configurations $(s_{i,j})_{i\geq 0, j\in J_i}$ defined in Section 5.2.1. This sequence of configurations is r-good for any finite r whenever the trace set of the LCS is unbounded. Conversely, if the sequence $(\pi_{i,j})_{i\geq 0, j\in J_i}$ is $(|Q|+1)$-good for the subword ordering $\preceq$, then $\mathcal{C}$ has an increasing fork. Indeed, let $r = |Q| + 1$; by definition of an r-good sequence, we can extract an increasing chain $\pi_{k_0} \preceq \pi_{k_1} \preceq \cdots \preceq \pi_{k_r}$

[1] We could first define a partial ordering $\leq$ on Π_M such that $(a+\varepsilon) \leq A^*$ whenever $a \in A$, and $A^* \leq B^*$ whenever $A \subseteq B$. The corresponding subword ordering (using $a_i \leq b_{f(i)}$ in its definition) would be equivalent to language inclusion, and result in *shorter* bad sequences.

of length $|Q|+1$ from the sequence $(\pi_{i,j})_{i\geq 0, j\in J_i}$. By Claim 24.1, this implies $L(\pi_{k_0}) \subseteq L(\pi_{k_1}) \subseteq \cdots \subseteq L(\pi_{k_r})$. Since $r = |Q|$, there exist $k_i < k_j$ such that $s_{k_i} = (q, \pi_{k_i})$ and $s_{k_j} = (q, \pi_{k_j})$ for some q in Q. Thus $s_{k_i} \leq s_{k_j}$, and we can apply Claim 21.2 to construct an increasing fork.

As the sequence $(\pi_{i,j})_{i\geq 0, j\in J_i}$ is $(\|.\|, g, 0)$-controlled by a primitive-recursive function according to Claim 24.2, the length of the sequence $(s_{i,j})_{i\geq 0, j\in J_i}$ needs not exceed $F^{|Q|}_{\omega^{|\Pi_M|-1}}(p(|Q|))$ for some primitive-recursive p by Fact 23, thus the upper bound of is multiply-recursive, and we obtain the desired $\mathbf{F}_{\omega^\omega}$ upper bound. □

6. Verifying trace bounded WSTS

As already mentioned in the introduction, liveness is generally undecidable for cd-WSTS. We show in this section that it becomes decidable for trace bounded systems obtained as the product of a cd-WSTS $\mathcal{S}$ with a deterministic Rabin automaton: we prove that it is decidable whether the language of ω-words of such a system is empty (Section 6.2) and apply it to the LTL model checking problem (Section 6.3). We conclude the section with a short survey on decidability issues when model checking WSTS (Section 6.4); but first we emphasize again the interest of trace boundedness for forward analysis techniques.

6.1. Forward analysis

Recall from the introduction that a forward analysis of the set of reachable states in an infinite LTS typically relies on *acceleration techniques* [see e.g. [4]] applied to loops w in Σ^*, provided one can effectively compute the effect of w^*. Computing the full reachability set (resp. coverability set for cd-WSTS) using a sequence $w_1^* \cdots w_n^*$ requires post* flattability (resp. cover flattability); however, as seen with Proposition 16 [resp. [2], Proposition 6], both these properties are already undecidable for cd-WSTS.

Trace bounded systems answer this issue since we can compute an appropriate finite sequence $w_1, \ldots, w_n$ and use it as acceleration sequence. Thus forward analysis techniques become complete for trace bounded systems. The Presburger accelerable counter systems of Demri et al. [18] are an example where, thanks to an appropriate representation for reachable states, the full reachability set is computable in the trace bounded case. In a more WSTS-centric setting, the forward Clover procedure of Finkel and Goubault-Larrecq for ∞-effective cd-WSTS terminates in the cover flattable case [2, Theorem 3], thus:

Corollary 25. *Let $\mathcal{S}$ be a trace bounded ∞-effective cd-WSTS. Then a finite representation of $\mathsf{Cover}_{\mathcal{S}}(s_0)$ can effectively be computed.*

Using the Cover set, one can answer state boundedness questions for WSTS. Furthermore, Cover sets and reachability sets coincide for lossy systems, and lossy channel systems in particular.

6.2. Deciding ω-language emptiness

ω-Regular languages. Let us recall the Rabin acceptance condition for ω-words (indeed, our restriction to deterministic systems demands a stronger condition than the Büchi one). Let us set some notation for infinite words in a labeled transition system $\mathcal{S} = \langle S, s_0, \Sigma, \to\rangle$. A sequence of states σ in S^ω is an *infinite execution* for the infinite word $a_0a_1\cdots$ in Σ^ω if $\sigma = s_0s_1\cdots$ with $s_i \xrightarrow{a_i} s_{i+1}$ for all i. We denote by $T_\omega(\mathcal{S})$ the set of infinite words that have an execution. The *infinity set* of an infinite sequence $\sigma = s_0s_1\cdots$ in S^ω is the set of symbols that appear infinitely often in σ: $\mathsf{inf}(\sigma) = \{s \in S \mid |\{i \in \mathbb{N} \mid s_i = s\}| = \omega\}$.

Let $\mathcal{S} = \langle S, s_0, \Sigma, \to, \leq\rangle$ be a deterministic WSTS and $\mathcal{A} = \langle Q, q_0, \Sigma, \delta\rangle$ a DFA. A *Rabin acceptance condition* is a finite set of pairs $(E_i, F_i)_i$ of finite subsets of Q. An infinite word w in Σ^ω is accepted by $\mathcal{S}\times\mathcal{A}$ if its infinite execution σ over $(S\times Q)^\omega$ verifies $\bigvee_i(\mathsf{inf}(\sigma)\cap(S\times E_i) = \emptyset \wedge \mathsf{inf}(\sigma)\cap(S\times F_i)\neq\emptyset)$. The set of accepted infinite words is denoted by $L_\omega(\mathcal{S}\times\mathcal{A}, (E_i, F_i)_i)$. Thus an infinite run is accepting if, for some i, it goes only finitely often through the states of E_i, but infinitely often through the states of F_i.

Deciding emptiness. We reduce the emptiness problem for $L_\omega(\mathcal{S}\times\mathcal{A}, (E_i, F_i)_i)$ to the trace boundedness problem for a finite set of cd-WSTS, which is decidable by Theorem 2. Remark that the following does not hold for nondeterministic systems, since any system can be turned into a trace bounded one by simply relabeling every transition with a single letter a.

Theorem 26. *Let $\mathcal{S}$ be an ∞-effective cd-WSTS, $\mathcal{A}$ a DFA, and $(E_i, F_i)_i$ a Rabin condition. If $\mathcal{S}\times\mathcal{A}$ is trace bounded, then it is decidable whether $L_\omega(\mathcal{S}\times\mathcal{A}, (E_i, F_i)_i)$ is empty.*

Proof. Set $\mathcal{S} = \langle S, s_0, \Sigma, \to, \leq\rangle$ and $\mathcal{A} = \langle Q, q_0, \Sigma, \delta\rangle$.

We first construct one cd-WSTS $\mathcal{S}_{i,1}$ for each condition (E_i, F_i) by adding to Σ a fresh symbol e_i, to $S\times Q$ the pairs (s, q_i) where s is in S and q_i is a fresh state for each q in E_i, and replace in $\to$ each transition $(s, q) \xrightarrow{a} (s', q')$ of $\mathcal{S}\times\mathcal{A}$

with q in E_i by two transitions $(s,q) \xrightarrow{e_i} (s,q_i) \xrightarrow{a} (s',q')$. Thus we read in $\mathcal{S}_i$ an e_i marker each time we visit some state in E_i.

Claim 26.1. *Each $\mathcal{S}_{i,1}$ is a trace bounded cd-WSTS.*

Proof of Claim 26.1. Observe that any trace of $\mathcal{S}_{i,1}$ is the image of a trace of $\mathcal{S} \times \mathcal{A}$ by a *generalized sequential machine* (GSM) $\mathcal{T}_i = \langle Q, q_0, \Sigma, \Sigma, \delta, \gamma \rangle$ using Σ both as input and output alphabet, and constructed from $\mathcal{A} = \langle Q, q_0, \Sigma, \delta \rangle$ with the same set of states and the same transitions, and by setting the output function γ from $Q \times \Sigma$ to Σ^* to be

$$(q,a) \mapsto e_i a \qquad \text{if } q \in E_i$$
$$(q,a) \mapsto a \qquad \text{otherwise.}$$

A GSM behaves like a DFA on a word $a_1 \cdots a_n$ by defining a run $q_0 \xrightarrow{a_1} q_1 \cdots q_{n-1} \xrightarrow{a_n} q_n$ with $q_{i+1} = \delta(q_i, a_{i+1})$ for all i, but additionally outputs the word $\gamma(q_0,a_1)\gamma(q_1,a_2)\cdots\gamma(q_{n-1},a_n)$, hence defining a function from finite words over its input alphabet to finite words over its output alphabet. Since bounded languages are closed under GSM mappings [1, Corollary on p. 348] and $\mathcal{S} \times \mathcal{A}$ is trace bounded, we know that $\mathcal{S}_{i,1}$ is trace bounded. □

In a second phase, we add a new symbol f_i and the elementary loops $(s,q) \xrightarrow{f_i} (s,q)$ for each (s,q) in $S \times F_i$ to obtain a system $\mathcal{S}_{i,2}$. Any run that visits some state in F_i has therefore the opportunity to loop on f_i^*.

In $\mathcal{S} \times \mathcal{A}$, visiting F_i infinitely often implies that we can find two configurations $(s,q) \leq (s',q)$ with q in F_i. In $\mathcal{S}_{i,2}$, we can thus recognize any sequence in $\{f_i, w\}^*$, where $(s,q) \xrightarrow{w} (s',q)$, from (s',q): $\mathcal{S}_{i,2}$ is not trace bounded.

Claim 26.2. *Each $\mathcal{S}_{i,2}$ is a cd-WSTS, and is trace unbounded iff there exists a run σ in $\mathcal{S} \times \mathcal{A}$ with $\mathsf{inf}(\sigma) \cap (S \times F_i) \neq \emptyset$.*

Proof of Claim 26.2. If there exists a run σ in $\mathcal{S} \times \mathcal{A}$ with $\mathsf{inf}(\sigma) \cap (S \times F_i) \neq \emptyset$, then we can consider the infinite sequence of visited states in $S \times F_i$ along σ. Since $\leq$ is a well quasi ordering on $S \times Q$, there exist two steps (s,q) and later (s',q') in this sequence with $(s,q) \leq (s',q')$. Observe that the same execution σ, modulo the transitions introduced in $\mathcal{S}_{i,1}$, is also possible in $\mathcal{S}_{i,2}$. Denote by w in Σ^* the sequence of transitions between these two steps, i.e. $(s,q) \xrightarrow{w} (s',q')$. By monotonicity of the transition relation of $\mathcal{S}_{i,2}$, we can recognize any sequence in $\{f_i, w\}^*$ from (q',s'). Thus $\mathcal{S}_{i,2}$ is not trace bounded.

Conversely, suppose that $\mathcal{S}_{i,2}$ is not trace bounded. By Lemma 11, it has an increasing fork with $(s_0,q_0) \xrightarrow{w} (s,q) \xrightarrow{au} (s_a,q)$ and $(s,q) \xrightarrow{bv} (s_b,q)$, $s_a \geq s$, $s_b \geq s$, $a \neq b$ in $\Sigma \uplus \{e_i, f_i\}$, u, w in $(\Sigma \uplus \{e_i, f_i\})_{\mathrm{acc}}$, and v in $(\Sigma \uplus \{e_i, f_i\})^*$.

Observe that if f_i only appears in the initial segment labeled by w, then a similar fork could be found in $\mathcal{S}_{i,1}$, since (s,q) would also be accessible. Thus, by Lemma 7, $\mathcal{S}_{i,1}$ would not be trace bounded. Therefore f_i appears in au or bv, and thus the corresponding runs for au or bv visit some state in F_i. But then, by monotonicity, we can construct a run that visits a state in F_i infinitely often. □

In the last, third step, we construct the synchronous product $\mathcal{S}_{i,3} = \mathcal{S}_{i,2} \times \mathcal{A}_i$, where $\mathcal{A}_i$ is a DFA for the language $(\Sigma \uplus \{e_i\})^* f_i (\Sigma \uplus \{f_i\})^*$ (where $\uplus$ denotes a disjoint union). This ensures that any run of $\mathcal{S}_{i,3}$ that goes through at least one f_i cannot go through e_i any longer, hence it visits the states in E_i only finitely many often. Since a run can always choose not to go through a f_i loop, the previous claim still holds. Therefore each $\mathcal{S}_{i,3}$ is a cd-WSTS, is trace unbounded iff there exists a run σ in $\mathcal{S} \times \mathcal{A}$ with $\mathsf{inf}(\sigma) \cap (S \times E_i) = \emptyset$ and $\mathsf{inf}(\sigma) \cap (S \times F_i) \neq \emptyset$, and we can apply Theorem 2. □

6.3. Model checking LTL formulæ

By standard automata-theoretic arguments [54,55], one can convert any linear-time temporal logic (LTL) formula φ over a finite set AP of atomic propositions, representing transition predicates, into a deterministic Rabin automaton $\mathcal{A}_{\neg\varphi}$ that recognizes exactly the runs over $\Sigma = 2^{\mathrm{AP}}$ that model $\neg\varphi$. The synchronized product of $\mathcal{A}_{\neg\varphi}$ with a complete, deterministic, ∞-effective, and trace bounded WSTS $\mathcal{S}$ is again trace bounded, and such that $L_\omega(\mathcal{S} \times \mathcal{A}, (E_i, F_i)_i) = T_\omega(\mathcal{S}) \cap L_\omega(\mathcal{A}, (E_i, F_i)_i)$. Theorem 26 entails that we can decide whether this language is empty, and whether all the infinite traces of $\mathcal{S}$ verify φ, noted $\mathcal{S} \models \varphi$. This reduction also works for LTL extensions that remain ω-regular.

Corollary 27. *Let $\mathcal{S} = \langle S, s_0, 2^{\mathrm{AP}}, \to, \leq \rangle$ be an ∞-effective trace bounded cd-WSTS, and φ a LTL formula on the set* AP *of atomic propositions. It is decidable whether $\mathcal{S} \models \varphi$.*

An alternative application of Theorem 26 is, rather than relying on the trace boundedness of $\mathcal{S}$, to ensure that $\mathcal{A}_{\neg\varphi}$ is trace bounded. To this end, the following slight adaptation of the flat counter logic of Comon and Cortier [56] is appropriate:

Definition 28. A LTL formula on a set AP of atomic propositions is *co-flat* if it is of form $\neg\varphi$, where φ follows the abstract syntax, where a stands for a letter in 2^{AP}:

$$\varphi ::= \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \mathsf{X}\varphi \mid \alpha \mathsf{U} \varphi \mid \mathsf{G}\alpha \qquad \text{(flat formulæ)}$$

$$\alpha ::= \bigwedge_{p \in a} p \wedge \bigwedge_{p \notin a} \neg p \,. \qquad \text{(alphabetic formulæ)}$$

In a conjunction $\varphi \wedge \varphi'$, one of φ or φ' could actually be an arbitrary LTL formula.

One can easily check that flat formulæ define languages of infinite words with bounded sets of finite prefixes, and we obtain:

Corollary 29. *Let $\mathcal{S} = \langle S, s_0, 2^{\mathrm{AP}}, \to, \leq \rangle$ be an ∞-effective cd-WSTS, and φ a co-flat LTL formula on the set* AP *of atomic propositions. It is decidable whether $\mathcal{S} \models \varphi$.*

Extensions of Corollary 29 to less restrictive LTL fragments seem possible, but our ideas thus far lead to rather unnatural conditions on the shape of formulæ.

6.4. Beyond ω-regular properties

We survey in this section some results from the model checking literature and their consequences for several classes of trace bounded WSTS. Outside the realm of ω-regular properties, we find essentially two kinds of properties: state-based properties or branching properties, or indeed a blend of the two [18,57,58].

Affine counter systems. Not all properties are decidable for trace bounded cd-WSTS, as seen with the following theorem on affine counter systems. Since these systems are otherwise completable, deterministic, and ∞-effective, action-based properties are decidable for them using Theorem 26, but we infer that state-based properties are undecidable for trace bounded ∞-effective cd-WSTS.

Theorem 30. *(See Cortier, 2002, [29].) Reachability is undecidable for trace bounded affine counter systems.*

Affine counter systems are thus the only class of systems (besides Minsky counter machines) in Fig. 2 for which trace boundedness does not yield a decidable reachability problem.

Presburger accelerable counter systems. Demri et al. [18] study the class of trace bounded counter systems for which accelerations can be expressed as Presburger relations.[2] Well-structured ∞-effective Presburger accelerable counter systems include trace bounded reset/transfer Petri nets and broadcast protocols, and Theorem 26 shows that ω-regular properties are decidable for them.

By the results of Demri et al., not only is the full reachability set computable for these systems, but furthermore an extension of state-based CTL* model checking with Presburger quantification on the paths is also decidable.

Guarded properties. Let us recall that state-based LTL model checking is already undecidable for Petri nets [28]. However, state-based properties become decidable for WSTS if they only allow to reason about upward-closed sets. This insight is applied by Bertrand and Schnoebelen [57], who define an upward and downward guarded fragment of state-based μ-calculus and prove its decidability for all WSTS. Goubault-Larrecq [58] presents a generalization to open sets in well topological spaces. Extensions of Theorem 26 along these lines could be investigated.

7. On trace unbounded WSTS

As many systems display some commutative behavior, and on that account fail to be trace bounded, Bardin et al. [4, Section 5.2] introduce *reductions* in order to enumerate the possible bounded expressions more efficiently, e.g. removal of identity loops, of useless conjugated sequences of transitions, and of commuting sequences. Such reductions are systematically looked for, up to some fixed length of the considered sequences.

Increasing forks suggest a different angle on this issue: whenever we identify a source of trace unboundedness, we could try to check whether the involved sequences commute, normalize our system, and restart the procedure on the new system, which is trace-equivalent modulo the spotted commutation. Considering again the example Petri net of Fig. 3, the two sequences c and d responsible for an increasing fork do commute. If we were to force any sequence of transitions in $\{c, d\}^*$ to be in the set $(cd)^*(c^* \cup d^*)$, then

[2] Whether trace boundedness is decidable for deterministic Presburger accelerable counter systems (i.e. not necessarily well-structured) is not currently known, while Proposition 15 answers negatively in the nondeterministic well-structured case.

- the set of reachable states would remain the same, but
- the normalized trace set would be

$$a^* \cup \bigcup_{0 \leq 2m \leq n} a^n b (cd)^m (c^{\leq n-2m} \cup d^{\leq n-2m}) ,$$

which is bounded.

Provided the properties to be tested do not depend on the relative order of c and d, we would now be able to apply Theorem 26.

We formalize this idea in Section 7.3 using a partial commutation relation (see Section 7.1 for background on partial commutations), and illustrate its interest for a bounded-session version of the Alternating Bit Protocol (see Section 7.2 for background on this protocol).

7.1. Partial commutations

Let Σ be a finite alphabet; a *dependence relation* $D \subseteq \Sigma \times \Sigma$ is a reflexive and symmetric relation on Σ. Its complement $I = (\Sigma \times \Sigma) \setminus D$ is an *independence relation*. On words in Σ^*, an independence relation can be interpreted as a congruence $\sim_I \subseteq \Sigma^* \times \Sigma^*$ generated by repeated applications of $ab \leftrightarrow_I ba$ for some (a, b) in I: $\sim_I = \leftrightarrow_I^*$, where $w \leftrightarrow_I w'$ if and only if there exist u and v in Σ^* and (a, b) in I with $w = uabv$ and $w' = ubav$. We work on infinite words modulo the partial commutations described by I.

Closure. The *limit extension* $\sim_I^{\lim} \subseteq \Sigma^\omega \times \Sigma^\omega$ of the congruence $\sim_I$ [59,60] is defined by $\sigma \sim_I^{\lim} \sigma'$ iff,

- for every finite prefix u of σ, there is a finite prefix u' of σ' and a finite word v of Σ^* such that $uv \sim_I u'$, and
- symmetrically, for every finite prefix u' of σ', there is a finite prefix u of σ and a finite word v' of Σ^* such that $u'v' \sim_I u$.

Consider for instance the relation $I \stackrel{\text{def}}{=} \{(a, b), (b, a)\}$; then $(aab)^\omega \sim_I^{\lim} (ab)^\omega$ (e.g. $(aab)^n b^n \sim_I (ab)^{2n}$ and $(ab)^n a^n \sim_I (aab)^n$), but $(aab)^\omega \not\sim_I^{\lim} a^\omega$ (e.g. $(aab)^n v \not\sim_I a^m$ for all $n > 0$, $m > 0$, and v in Σ^*).

A language $L \subseteq \Sigma^*$ (resp. $L \subseteq \Sigma^\omega$) is *$I$-closed*, if for any σ in L, and for every σ' with $\sigma \sim_I \sigma'$ (resp. $\sigma \sim_I^{\lim} \sigma'$), σ' is also in L. The closure of an ω-regular language for a given partial commutation is decidable, and more precisely PSPACE-complete if the language is given as a Büchi automaton or an LTL formula [60].

Definition 31. An LTS is *I-diamond* if, for any pair (a, b) of I, and for any states s in $\mathsf{dom} \xrightarrow{ab} \cap \mathsf{dom} \xrightarrow{ba}$ and s' in S, $s \xrightarrow{ab} s'$ iff $s \xrightarrow{ba} s'$.

We have the following sufficient condition for the closure of $T_\omega(\mathcal{S})$, which is decidable for I-diamond WSTS: just compare the elements in the finite bases for $\mathsf{dom} \xrightarrow{ab}$ and $\mathsf{dom} \xrightarrow{ba}$.

Lemma 32. *Let I be an independence relation and $\mathcal{S}$ an LTS, both on Σ. If $\mathcal{S}$ is I-diamond and, for all (a, b) of I, $\mathsf{dom} \xrightarrow{ab} = \mathsf{dom} \xrightarrow{ba}$, then $T_\omega(\mathcal{S})$ is I-closed.*

Proof. One can easily check that this condition implies that the set of finite traces $T(\mathcal{S})$ is I-closed.

Let now σ be an infinite word in $T_\omega(\mathcal{S})$, and σ' an infinite word in Σ^ω with $\sigma \sim_I^{\lim} \sigma'$, but suppose that σ' is not in $T_\omega(\mathcal{S})$. Thus there exists a finite prefix u' of σ' that does not belong to $T(\mathcal{S})$. By definition of $\sim_I^{\lim}$, there is however a prefix u of σ and a word v' of Σ^* such that $u'v' \sim_I u$. But this contradicts the closure of $T(\mathcal{S})$, since u is in $T(\mathcal{S})$, but $u'v'$ is not—or u' would be in the prefix-closed language $T(\mathcal{S})$. □

However, already in the case of I-diamond WSTS and already for finite traces, I-closure is undecidable; a sufficient condition like Lemma 32 is the best we can hope for.

Proposition 33. *Let I be an independence relation and $\mathcal{S}$ an I-diamond cd-WSTS, both on Σ. It is undecidable whether $T(\mathcal{S})$ is I-closed or not.*

Proof. We reduce the (undecidable) reachability problem for a transfer Petri net $\mathcal{N}$ and a marking m [40] to the I-closure problem for a new transfer Petri net $\mathcal{N}'$. Let us recall that a *transfer arc* (p, t, p') transfers all the tokens from a place p to another place p' when t is fired.

The new transfer Petri net $\mathcal{N}'$ extends $\mathcal{N}$ with three new places sim, sum, and test, and three new transitions t, a, and b (see Fig. 11). Its initial marking is expanded so that sim originally contains one token, sum the sum $s_{m_0} = \sum_p m_0(p)$ of all

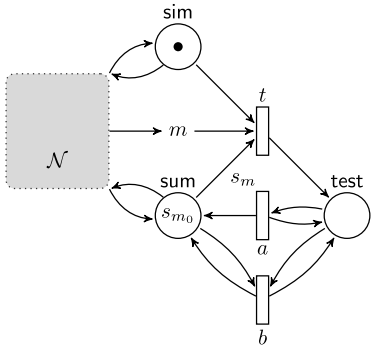

Fig. 11. The transfer Petri net $\mathcal{N}'$ of the proof of Proposition 33.

the tokens in the initial marking of $\mathcal{N}$, and test no token. It simulates $\mathcal{N}$ while a token resides in sim, and updates sum so that it contains at all times the sum of the tokens in all the places of $\mathcal{N}$. Transfer arcs are not an issue since they do not change this overall sum of tokens. Nondeterministically, $\mathcal{N}'$ fires t, which removes $m(p)$ in each place p of $\mathcal{N}$, one token from sim, $s_m = \sum_p m(p)$ tokens from sum, and places one token in test.

Now, a token can appear in test if and only if a marking m' larger than m can be reached in $\mathcal{N}'$. Furthermore, the distance $\sum_p m'(p) - m(p)$ is in sum, so that m is reachable in $\mathcal{N}$ if and only if a marking with one token in test and no token in sum is reachable in $\mathcal{N}'$.

The latter condition is tested by having a remove one token from test and put one token in sum and one back in test, and b remove one from sum and test and put them back. Set $I \stackrel{\text{def}}{=} \{(a,b),(b,a)\}$; $\mathcal{N}'$ is I-diamond. The transition sequence ab can be fired if and only if there if a token in test, but ba further requires sum not to be empty. Thus a and b do not commute if and only if m is reachable in $\mathcal{N}$. □

Foata normal form. Let us assume an arbitrary linear ordering $<$ on Σ. For an independence relation I, we denote by $\mathcal{C}(I)$ the set of cliques of I, i.e.

$$\mathcal{C}(I) \stackrel{\text{def}}{=} \{C \subseteq \Sigma \mid \forall a, b \in C, (a,b) \in I\}\ .$$

We further introduce a homomorphism $\nu : 2^\Sigma \to \Sigma^*$ by

$$\nu(\{a_1, a_2, \ldots, a_k\}) = a_1 a_2 \cdots a_k \qquad \text{if } a_1 < a_2 < \cdots < a_k.$$

An infinite word σ in Σ^ω is in *Foata normal form* [see e.g. [61]] if there is an infinite decomposition $\sigma = \nu(C_0)\nu(C_1)\cdots$ with each C_i in $\mathcal{C}(I)$, and for each a in C_i, there exists b in C_{i-1} such that (a,b) is in D. As indicated by its name, for any word σ in Σ^ω, there exists a unique word $\mathsf{fnf}_I(\sigma)$ in Foata normal form such that $\sigma \sim_I^{\lim} \mathsf{fnf}_I(\sigma)$. For instance $\mathsf{fnf}_I((aab)^\omega) = (ab)^\omega$ for $I = \{(a,b),(b,a)\}$.

Let us finally define the *normalizing language* N_I of I as the set of all infinite words in Foata normal form. The following lemma shows that N_I is very well behaved, being recognized by a deterministic Büchi automaton $\mathcal{B}_I$ with only accepting states. Thus its synchronous product with a WSTS $\mathcal{S}$ does not require the addition of an acceptance condition: $T_\omega(\mathcal{S} \times \mathcal{B}_I) = T_\omega(\mathcal{S}) \cap N_I$.

Lemma 34. *Let I be an independence relation on Σ. Then N_I is a topologically closed ω-regular language.*

Proof. The topologically closed ω-regular languages, aka "safety" languages, are the languages recognized by finite deterministic Büchi automata with only accepting states. We provide such an automaton $\mathcal{B}_I = \langle Q, \Sigma, q_0, \delta, Q \rangle$ such that $L(\mathcal{B}_I) = N_I$.

Set $Q \stackrel{\text{def}}{=} \{q_0\} \cup (\mathcal{C}(I) \cup \{\Sigma\}) \times \mathcal{C}(I) \times \Sigma$. We define $\delta(q_0, a)$ as $(\Sigma, \{a\}, a)$ for all a in Σ; for all C_1 in $\mathcal{C}(I) \cup \{\Sigma\}$, C_2 in $\mathcal{C}(I)$, a, b in Σ, we define $\delta((C_1, C_2, a), b)$ by

$$\begin{cases} (C_1, C_2 \cup \{b\}, b) & \text{if } a < b,\ \exists d \in C_1, (b,d) \in D, \\ & \quad \text{and } \forall d \in C_2, (b,d) \in I, \\ (C_2, \{b\}, b) & \text{if } \exists d \in C_2, (b,d) \in D\ . \end{cases}$$

The automaton simultaneously checks that consecutive cliques enforce the Foata normal form, and that the individual letters of each clique are ordered according to $<$. □

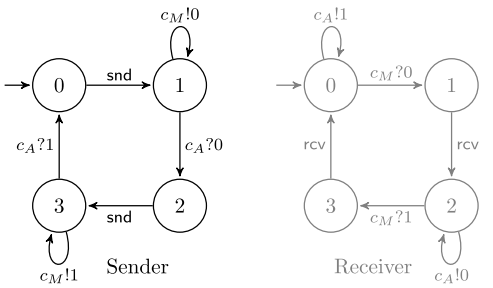

Fig. 12. The Alternating Bit Protocol.

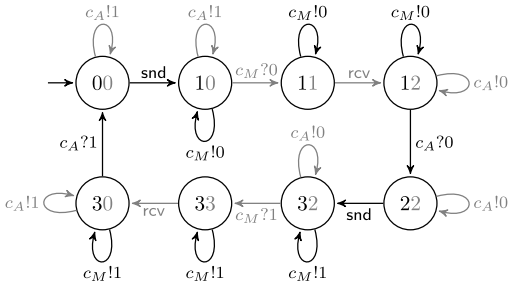

Fig. 13. Synchronized view of the ABP.

7.2. The Alternating Bit Protocol

The *Alternating Bit Protocol* (ABP) is one of the oldest case studies [62]. It remains interesting today because no complete and automatic procedure exists for its verification. It can be nicely modeled as a lossy channel system [see [27], and the next discussion "A Quick Tour"], but even in this representation, liveness properties cannot be checked. We believe it provides a good illustration of the kind of issues that make a system trace unbounded, which we categorize into commutativity issues, which we tackle through normalization, and main control loop issues, which we avoid by bounding the number of sessions.

A quick tour. If the ABP is modeled as a fifo automaton (in fact two finite automata communicating through two fifo queues), then all non-trivial properties are undecidable, because fifo automata can simulate Turing machines [see e.g. [63]]. Nevertheless, several classes of fifo automata have been studied in the literature, often with decidable reachability problems:

- One may observe that for any control state q of this particular fifo automaton, the language of the two fifo queues is *recognizable* (as a subset of $\{q\} \times A^* \times B^*$ where A and B are the alphabets of the queues). Pachl [64] has shown that reachability and safety are then decidable. But this recognizability property itself is in general undecidable.
- One may also observe that the languages of the fifo queues contents are *bounded* [65], and then one may simulate the fifo automaton with a Petri net and decide reachability. Again, this subclass of fifo automata is not recursive.
- Yet another way is to use *loop acceleration* with QDDs [66] or more generally CQDDs [67] as symbolic representations, and to observe that the reachability set is CQDD computable; but still without termination guarantee when applied to non-flat systems.

Neither of these techniques is fully automatic nor allows to check liveness properties.

The most effective approach is arguably to model the ABP as a lossy channel system (see Fig. 12); reachability and safety are then decidable, but liveness remains undecidable. Furthermore, a forward analysis using SREs as symbolic representations—as performed by a tool like TReX—will terminate and construct a finite symbolic graph (for the verification of safety properties) [27]: indeed, the ABP is *cover flattable*, but unfortunately this property is in general undecidable.

Verification. We model the ABP as two functional lossy channel systems (Sender and Receiver) that run in parallel, and communicate through two shared channels c_M for messages and c_A for acknowledgments. Our correctness property is whether each sent message (proposition snd) is eventually received (proposition rcv):

$$\mathsf{G}(\mathsf{snd} \Rightarrow \mathsf{X}(\neg\, \mathsf{snd}\ \mathsf{U}\ \mathsf{rcv}))\,, \qquad (\varphi_{\mathrm{ABP}})$$

under a weak fairness assumption (every continuously firable transition is eventually fired).

The full system is displayed for its useful accessible part in Fig. 13, with Receiver's transitions in grey. This system is clearly not trace bounded, thus we cannot apply Theorem 26 alone.

Table 2
Some decidability results for selected classes of cd-WSTS—Petri nets (PN), affine counter systems (ACS), and functional lossy channel systems (LCS)—in the general and trace bounded cases (t.b.).

	PN	t.b. PN	ACS	t.b. ACS	LCS	t.b. LCS
Reachability	Yes	Yes	No	No	Yes	Yes
Post* inclusion	No	Yes	No	No	No	Yes
Liveness	Yes	Yes	No	Yes	No	Yes

7.3. Trace bounded modulo I

The search for increasing forks on the ABP successively finds four witnesses of trace unboundedness in states 10, 12, 32, and 30, where at each occasion two competing elementary loops can be fired. Thankfully, all these loops commute, because they involve two different channels. Our goal is to transform our system in order to remove these forks, while maintaining the ability to verify (φ_{ABP}).

Definition 35. A WSTS $\mathcal{S}$ is *trace bounded modulo* I an independence relation, if $T_\omega(\mathcal{S})$ is I-closed and the set of finite prefixes of the normalized language $T_\omega(\mathcal{S}) \cap N_I$ is trace bounded.

By Lemma 34, we can construct a cd-WSTS $\mathcal{S}'$ for $T_\omega(\mathcal{S}) \cap N_I$, and decide whether it is trace bounded thanks to Theorem 2. Thus trace boundedness modulo I is decidable for I-closed WSTS.

Finally, provided the language $L(\neg\varphi)$ of the property to verify is also I-closed, the normalized system and the original system are equivalent when it comes to verifying φ. Indeed, we can generalize Theorem 26 to trace bounded modulo I cd-WSTS and I-closed ω-regular languages:

Theorem 36. *Let I be an independence relation, $\mathcal{S}$ be a trace bounded modulo I cd-WSTS, and L an I-closed ω-regular language, all three on Σ. Then it is decidable whether $T_\omega(\mathcal{S}) \cap L$ is empty.*

Proof. By Lemma 34, we can construct a cd-WSTS $\mathcal{S}'$ for $T_\omega(\mathcal{S}) \cap N_I$, which will be trace bounded by hypothesis. Wlog., we can assume that we have a DFA with a Rabin acceptance condition for L, and can apply Theorem 26 to decide whether $T_\omega(\mathcal{S}') \cap L = \emptyset$.

It remains to prove that

$$T_\omega(\mathcal{S}) \cap L = \emptyset \text{ iff } T_\omega(\mathcal{S}') \cap L = \emptyset\,.$$

Obviously, if $T_\omega(\mathcal{S}) \cap L$ is empty, then the same holds for $T_\omega(\mathcal{S}') \cap L$. For the converse, let σ be a word in $T_\omega(\mathcal{S}) \cap L$. Then, since $\mathcal{S}$ is I-closed, $\mathsf{fnf}_I(\sigma)$ also belongs to $T_\omega(\mathcal{S})$ and to N_I, and thus to $T_\omega(\mathcal{S}')$. And because L is I-closed, $\mathsf{fnf}_I(\sigma)$ further belongs to L, hence to $T_\omega(\mathcal{S}') \cap L$. □

Once our system is normalized against partial commutations, the only remaining source of trace unboundedness is the main control loop. By bounding the number of sessions of the protocol, i.e. by unfolding this main control loop a bounded number of times, we obtain a trace bounded system.

This transformation would disrupt the verification of (φ_{ABP}), if it were not for the two following observations:

1. The full set of all reachable configurations is already explored after two traversals of the main control loop. This is established automatically thanks to Corollary 25 on the 2-unfolding of the normalized ABP, which is a trace bounded cd-WSTS. Thus any possible session, with any possible reachable initial configuration, can already be exhibited at the second traversal of the system.
2. Our property (φ_{ABP}) is intra-session: it only requires to be tested against any possible session.

The overall approach, thanks to the concept of trace boundedness modulo partial commutations, thus succeeds in reducing the ABP to a trace bounded system where our liveness property can be verified.

8. Boundedness is not a Weakness

To paraphrase the title *Flatness is not a Weakness* [56], trace boundedness is a powerful property for the analysis of systems, as demonstrated with the termination of forward analyses and the decidability of ω-regular properties for trace bounded WSTS (see also Table 2)—and is implied by flatness. More examples of its interest can be found in the recent literature on the verification of multithreaded programs, where trace boundedness of the context-free synchronization languages yields decidable reachability [68,69].

Most prominently, trace boundedness has the considerable virtue of being decidable for a large class of systems, the ∞-effective complete deterministic WSTS. There is furthermore a range of unexplored possibilities beyond partial commutations (starting with semi-commutations or contextual commutations) that could help turn a system into a trace bounded one.

Acknowledgements

We thank the anonymous reviewers for their careful reading, which improved the paper.

References

[1] S. Ginsburg, E.H. Spanier, Bounded Algol-like languages, Trans. Amer. Math. Soc. 113 (2) (1964) 333–368, http://dx.doi.org/10.2307/1994067.
[2] A. Finkel, J. Goubault-Larrecq, Forward analysis for WSTS, Part II: Complete WSTS, Logical Methods in Computer Science 8 (3), http://dx.doi.org/10.2168/LMCS-8(3:28)2012.
[3] B. Boigelot, P. Wolper, Symbolic verification with periodic sets, in: CAV'94, in: Lecture Notes in Computer Science, vol. 818, Springer, 1994, pp. 55–67.
[4] S. Bardin, A. Finkel, J. Leroux, Ph. Schnoebelen, Flat acceleration in symbolic model checking, in: ATVA 2005, in: Lecture Notes in Computer Science, vol. 3707, Springer, 2005, pp. 474–488.
[5] A. Annichini, A. Bouajjani, M. Sighireanu, TReX: a tool for reachability analysis of complex systems, in: CAV 2001, in: Lecture Notes in Computer Science, vol. 2102, Springer, 2001, pp. 368–372.
[6] Lash, The Liège automata-based symbolic handler, http://www.montefiore.ulg.ac.be/~boigelot/research/lash/, 2001.
[7] S. Bardin, A. Finkel, J. Leroux, L. Petrucci, Fast: acceleration from theory to practice, Int. J. Softw. Tools Technol. Transf. 10 (5) (2008) 401–424, http://dx.doi.org/10.1007/s10009-008-0064-3.
[8] P.A. Abdulla, K. Čerāns, B. Jonsson, Y.-K. Tsay, Algorithmic analysis of programs with well quasi-ordered domains, Inform. and Comput. 160 (1–2) (2000) 109–127, http://dx.doi.org/10.1006/inco.1999.2843.
[9] T.A. Henzinger, O. Kupferman, S. Qadeer, From *Pre*-historic to *Post*-modern symbolic model checking, Form. Methods Syst. Des. 23 (3) (2003) 303–327, http://dx.doi.org/10.1023/A:1026228213080.
[10] P.A. Abdulla, B. Jonsson, Verifying programs with unreliable channels, Inform. and Comput. 127 (2) (1996) 91–101, http://dx.doi.org/10.1006/inco.1996.0053.
[11] J. Leroux, G. Sutre, On flatness for 2-dimensional vector addition systems with states, in: CONCUR 2004, in: Lecture Notes in Computer Science, vol. 3170, Springer, 2004, pp. 402–416.
[12] J. Leroux, G. Sutre, Flat counter automata almost everywhere!, in: ATVA 2005, in: Lecture Notes in Computer Science, vol. 3707, Springer, 2005, pp. 489–503.
[13] M. Blondin, A. Finkel, S. Göller, C. Haase, P. McKenzie, Reachability in two-dimensional vector addition systems with states is PSPACE-complete, in: LICS 2015, ACM Press, 2015, pp. 32–43.
[14] A. Finkel, J. Goubault-Larrecq, Forward analysis for WSTS, Part I: completions, in: STACS 2009, in: Leibniz International Proceedings in Informatics, vol. 3, LZI, 2009, pp. 433–444.
[15] G. Geeraerts, J.-F. Raskin, L. Van Begin, Expand, Enlarge and Check: new algorithms for the coverability problem of WSTS, J. Comput. System Sci. 72 (1) (2006) 180–203, http://dx.doi.org/10.1016/j.jcss.2005.09.001.
[16] R.M. Karp, R.E. Miller, Parallel program schemata, J. Comput. System Sci. 3 (2) (1969) 147–195, http://dx.doi.org/10.1016/S0022-0000(69)80011-5.
[17] A. Finkel, Reduction and covering of infinite reachability trees, Inform. and Comput. 89 (2) (1990) 144–179, http://dx.doi.org/10.1016/0890-5401(90)90009-7.
[18] S. Demri, A. Finkel, V. Goranko, G. van Drimmelen, Model-checking CTL* over flat Presburger counter systems, J. Appl. Non-Classical Logics 20 (4) (2011) 313–344, http://dx.doi.org/10.3166/jancl.20.313-344.
[19] S. Demri, A. Kumar Dhar, A. Sangnier, Taming past LTL and flat counter systems, Inform. and Comput. 242 (2015) 306–339, http://dx.doi.org/10.1016/j.ic.2015.03.007.
[20] P. Ganty, R. Iosif, Interprocedural reachability for flat integer programs, in: FCT 2015, in: Lecture Notes in Computer Science, vol. 9210, Springer, 2015, pp. 133–145.
[21] R. Siromoney, A characterization of semilinear sets, Proc. Amer. Math. Soc. 21 (3) (1969) 689–694, http://dx.doi.org/10.1090/S0002-9939-1969-0239891-9.
[22] P. Chambart, Ph. Schnoebelen, The ordinal recursive complexity of lossy channel systems, in: LICS 2008, IEEE, 2008, pp. 205–216.
[23] M. Blockelet, S. Schmitz, Model-checking coverability graphs of vector addition systems, in: MFCS 2011, in: Lecture Notes in Computer Science, vol. 6907, Springer, 2011, pp. 108–119.
[24] P.A. Abdulla, B. Jonsson, Undecidable verification problems for programs with unreliable channels, Inform. and Comput. 130 (1) (1996) 71–90, http://dx.doi.org/10.1006/inco.1996.0083.
[25] R. Mayr, Undecidable problems in unreliable computations, Theoret. Comput. Sci. 297 (1–3) (2003) 337–354, http://dx.doi.org/10.1016/S0304-3975(02)00646-1.
[26] E.A. Emerson, K.S. Namjoshi, On model checking for non-deterministic infinite-state systems, in: LICS'98, IEEE, 1998, pp. 70–80.
[27] P.A. Abdulla, A. Collomb-Annichini, A. Bouajjani, B. Jonsson, Using forward reachability analysis for verification of lossy channel systems, Form. Methods Syst. Des. 25 (1) (2004) 39–65, http://dx.doi.org/10.1023/B:FORM.0000033962.51898.1a.
[28] J. Esparza, Decidability of model checking for infinite-state concurrent systems, Acta Inform. 34 (2) (1997) 85–107, http://dx.doi.org/10.1007/s002360050074.
[29] V. Cortier, About the decision of reachability for register machines, RAIRO Theor. Inform. Appl. 36 (4) (2002) 341–358, http://dx.doi.org/10.1051/ita:2003001.
[30] C. Dufourd, P. Jančar, Ph. Schnoebelen, Boundedness of reset P/T nets, in: ICALP'99, in: Lecture Notes in Computer Science, vol. 1644, Springer, 1999, pp. 301–310.
[31] M. Krohn, E. Kohler, M.F. Kaashoek, Events can make sense, in: USENIX 2007, 2007, pp. 87–100.
[32] P. Ganty, R. Majumdar, Algorithmic verification of asynchronous programs, ACM Trans. Program. Lang. Syst. 34 (1) (2012) 6:1–6:48, http://dx.doi.org/10.1145/2160910.2160915.
[33] A. Finkel, Ph. Schnoebelen, Well-structured transition systems everywhere!, Theoret. Comput. Sci. 256 (1–2) (2001) 63–92, http://dx.doi.org/10.1016/S0304-3975(00)00102-X.
[34] G. Geeraerts, J. Raskin, L.V. Begin, Well-structured languages, Acta Inform. 44 (3–4) (2007) 249–288, http://dx.doi.org/10.1007/s00236-007-0050-3.
[35] M.L. Minsky, Computation: Finite and Infinite Machines, Prentice-Hall, Inc., 1967.

[36] A. Finkel, P. McKenzie, C. Picaronny, A well-structured framework for analysing Petri net extensions, Inform. and Comput. 195 (1–2) (2004) 1–29, http://dx.doi.org/10.1016/j.ic.2004.01.005.
[37] Ph. Schnoebelen, Lossy counter machines decidability cheat sheet, in: RP 2010, in: Lecture Notes in Computer Science, vol. 6227, Springer, 2010, pp. 51–75.
[38] A. Finkel, J. Leroux, How to compose Presburger-accelerations: applications to broadcast protocols, in: FSTTCS 2002, in: Lecture Notes in Computer Science, vol. 2556, Springer, 2002, pp. 145–156.
[39] J. Esparza, A. Finkel, R. Mayr, On the verification of broadcast protocols, in: LICS'99, IEEE, 1999, pp. 352–359.
[40] C. Dufourd, A. Finkel, Ph. Schnoebelen, Reset nets between decidability and undecidability, in: ICALP'98, in: Lecture Notes in Computer Science, vol. 1443, Springer, 1998, pp. 103–115.
[41] M.-F. Atig, P. Habermehl, On Yen's path logic for Petri nets, Internat. J. Found. Comput. Sci. 22 (4) (2011) 783–799, http://dx.doi.org/10.1142/S0129054111008428.
[42] S. Schmitz, Complexity hierarchies beyond Elementary, ACM Trans. Comput. Theory 8 (1) (2016) 1–36, http://dx.doi.org/10.1145/2858784.
[43] M. Löb, S. Wainer, Hierarchies of number theoretic functions, I, Arch. Math. Logic 13 (1970) 39–51, http://dx.doi.org/10.1007/BF01967649.
[44] R. Lipton, The reachability problem requires exponential space, Tech. Rep. 62, Yale University, 1976.
[45] Ph. Schnoebelen, Revisiting Ackermann-hardness for lossy counter systems and reset Petri nets, in: MFCS 2010, in: Lecture Notes in Computer Science, vol. 6281, 2010, pp. 616–628.
[46] E. Cardoza, R.J. Lipton, A.R. Meyer, Exponential space complete problems for Petri nets and commutative semigroups, in: STOC'76, ACM Press, 1976, pp. 50–54.
[47] C. Rackoff, The covering and boundedness problems for vector addition systems, Theoret. Comput. Sci. 6 (2) (1978) 223–231, http://dx.doi.org/10.1016/0304-3975(78)90036-1.
[48] J. Esparza, Decidability and complexity of Petri net problems – an introduction, in: Lectures on Petri Nets I: Basic Models, in: Lecture Notes in Computer Science, vol. 1491, Springer, 1998, pp. 374–428.
[49] M. Jantzen, Complexity of Place/Transition nets, in: Petri Nets: Central Models and Their Properties, in: Lecture Notes in Computer Science, vol. 254, Springer, 1987, pp. 413–434.
[50] P. Gawrychowski, D. Krieger, N. Rampersad, J. Shallit, Finding the growth rate of a regular or context-free language in polynomial time, Internat. J. Found. Comput. Sci. 21 (4) (2010) 597–618, http://dx.doi.org/10.1142/S0129054110007441.
[51] P. Habermehl, R. Mayr, A note on SLRE, URL http://homepages.inf.ed.ac.uk/rmayr/slre.ps.gz, 2000.
[52] D. Figueira, S. Figueira, S. Schmitz, Ph. Schnoebelen, Ackermannian and primitive-recursive bounds with Dickson's Lemma, in: LICS 2011, IEEE, 2011, pp. 269–278.
[53] S. Schmitz, Ph. Schnoebelen, Multiply-recursive upper bounds with Higman's Lemma, in: ICALP 2011, in: Lecture Notes in Computer Science, vol. 6756, Springer, 2011, pp. 441–452.
[54] M.Y. Vardi, P. Wolper, An automata-theoretic approach to automatic program verification, in: LICS'86, 1986, pp. 332–344.
[55] S. Safra, On the complexity of ω-automata, in: FOCS'88, IEEE, 1988, pp. 319–327.
[56] H. Comon, V. Cortier, Flatness is not a weakness, in: CSL 2000, in: Lecture Notes in Computer Science, vol. 1862, Springer, 2000, pp. 262–276.
[57] N. Bertrand, Ph. Schnoebelen, Computable fixpoints in well-structured symbolic model checking, Form. Methods Syst. Des. 43 (2) (2013) 233–267, http://dx.doi.org/10.1007/s10703-012-0168-y.
[58] J. Goubault-Larrecq, On Noetherian spaces, in: LICS 2007, IEEE, 2007, pp. 453–462.
[59] V. Diekert, P. Gastin, A. Petit, Rational and recognizable complex trace languages, Inform. and Comput. 116 (1) (1995) 134–153, http://dx.doi.org/10.1006/inco.1995.1010.
[60] D. Peled, T. Wilke, P. Wolper, An algorithmic approach for checking closure properties of temporal logic specifications and ω-regular languages, Theoret. Comput. Sci. 195 (2) (1998) 183–203, http://dx.doi.org/10.1016/S0304-3975(97)00219-3.
[61] P. Gastin, A. Petit, Infinite traces, in: The Book of Traces, World Scientific Publishing, 1995, pp. 393–486, Chap. 11.
[62] G.v. Bochmann, C.A. Sunshine, Formal methods in system design, IEEE Trans. Commun. 28 (4) (1980) 624–631, http://dx.doi.org/10.1109/TCOM.1980.1094685.
[63] D. Brand, P. Zafiropulo, On communicating finite-state machines, J. ACM 30 (2) (1983) 323–342, http://dx.doi.org/10.1145/322374.322380.
[64] J. Pachl, Reachability problems for communicating finite state machines, Tech. Rep. CS-82-12, University of Waterloo, Department of Computer Science, URL http://arxiv.org/abs/cs/0306121, 1982.
[65] A. Finkel, A. Choquet, Simulation of linear fifo nets by Petri nets having a structured set of terminal markings, in: APN'87, 1987.
[66] B. Boigelot, P. Godefroid, Symbolic verification of communication protocols with infinite state spaces using QDDs, Form. Methods Syst. Des. 14 (3) (1999) 237–255, http://dx.doi.org/10.1023/A:1008719024240.
[67] A. Bouajjani, P. Habermehl, Symbolic reachability analysis of FIFO-channel systems with nonregular sets of configurations, Theoret. Comput. Sci. 221 (1–2) (1999) 211–250, http://dx.doi.org/10.1016/S0304-3975(99)00033-X.
[68] V. Kahlon, Tractable dataflow analysis for concurrent programs via bounded languages, Patent Application Publication US 2009/0193417 A1, 2009.
[69] P. Ganty, R. Majumdar, B. Monmege, Bounded underapproximations, Form. Methods Syst. Des. 40 (2) (2012) 206–231, http://dx.doi.org/10.1007/s10703-011-0136-y.