Verification of Flat FIFO Systems

Alain Finkel
LSV, ENS Paris-Saclay, CNRS, Université Paris-Saclay, France
UMI ReLaX, French-Indian research laboratory in computer sciences, Chennaï, India

M. Praveen
Chennai Mathematical Institute, India
UMI ReLaX, French-Indian research laboratory in computer sciences, Chennaï, India

Abstract

The decidability and complexity of reachability problems and model-checking for flat counter systems have been explored in detail. However, only few results are known for flat FIFO systems, only in some particular cases (a single loop or a single bounded expression). We prove, by establishing reductions between properties, and by reducing SAT to a subset of these properties that many verification problems like reachability, non-termination, unboundedness are NP-complete for flat FIFO systems, generalizing similar existing results for flat counter systems. We construct a trace-flattable counter system that is bisimilar to a given flat FIFO system, which allows to model-check the original flat FIFO system. Our results lay the theoretical foundations and open the way to build a verification tool for (general) FIFO systems based on analysis of flat subsystems.

2012 ACM Subject Classification Theory of computation → Parallel computing models

Keywords and phrases Infinite state systems, FIFO, counters, flat systems, reachability, termination, complexity

Digital Object Identifier 10.4230/LIPIcs.CONCUR.2019.12

Funding The work reported was carried out in the framework of ReLaX, UMI2000 (ENS Paris-Saclay, CNRS, Univ. Bordeaux, CMI, IMSc). This work was also supported by the grant ANR-17-CE40-0028 of the French National Research Agency ANR (project BRAVAS).
M. Praveen: Partially supported by a grant from the Infosys foundation.

1 Introduction

FIFO systems. Asynchronous distributed processes communicating through First In First Out (FIFO) channels are used since the seventies as models for protocols [33], distributed and concurrent programming and more recently for web service choreography interface [12]. Since FIFO systems simulate counter machines, most reachability properties are *undecidable* for FIFO systems: for example, the basic task of checking if the number of messages buffered in a channel can grow unboundedly is undecidable [11].

There aren't many interesting and useful FIFO subclasses with a *decidable* reachability problem. Considering FIFO systems with a unique FIFO channel is not a useful restriction since they may simulate Turing machines [11]. A few examples of decidable subclasses are half-duplex systems [13] (but they are restricted to two machines since the natural extension to three machines leads to undecidability), existentially bounded deadlock free FIFO systems [26] (but it is undecidable to check if a system is existentially bounded, even for deadlock free FIFO systems), synchronisable FIFO systems (the property of synchronisability is undecidable [24] and moreover, it is not clear which properties of synchronisable systems are decidable), flat FIFO systems [6, 7] and lossy FIFO systems [1] (but one loses the perfect FIFO mechanism).

Flat systems. A flat system [4, 23, 14, 5] is a system with a finite control structure such that every control-state belongs to at most one loop. Equivalently, the language of the control structure is included in a bounded language of the form $w_1^* w_2^* ... w_k^*$ where every w_i is a non empty word. Analyzing flat systems essentially reduces to accelerating loops (i.e., to compute finite representations of the effect of iterating each loop arbitrarily many times) and to connect these finite representations with one another. Flat systems are particularly interesting since one may under-approximate any system by its flat subsystems.

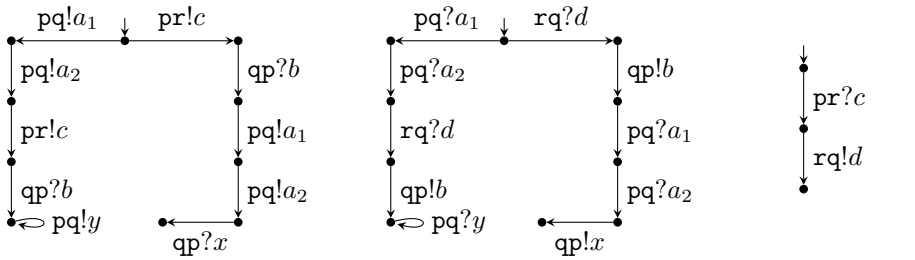
For counter systems [19, 28], this strategy lead to some tools like FAST [3], LASH, TREX [2], FLATA [10] which enumerate all flat subsystems till the reachability set is reached. This strategy is not an algorithm since it may never terminate on some inputs. However in practice, it terminates in many cases; e.g., in [3], 80% of the examples (including Petri nets and multi-threaded Java programs) could be effectively verified. The complexity of flat counter systems is well-known: reachability is NP-complete for variations of flat counter systems [27, 9, 18], model-checking first-order formulae and linear μ-calculus formulae is PSPACE-complete while model-checking Büchi automata is NP-complete [17]; equivalence between model-checking flat counter systems and Presburger arithmetic is established in [16].

Flat FIFO systems. We know almost nothing about flat FIFO systems, even the complexity of reachability is not known. Boigelot et al. [6] used recognizable languages (QDD) for representing FIFO channel contents and proved that the acceleration of *one-counting* loops (a loop is one-counting if it sends messages to only one channel), from an initial QDD, produces another computable QDD. Bouajjani and Habermehl [7] proved that the acceleration of *any* loop can be finitely represented by combining a deterministic flat finite automaton and a Presburger formula (CQDD) that are both computable. However, surprisingly, no upper bound for the Boigelot et al.'s and for the Bouajjani et al.'s loop-acceleration algorithms are known. Just the complexity of the inclusion problem for QDD, CQDD and SLRE (SLRE are both QDD and CQDD) are partially known (respectively PSPACE-complete, N2EXPTIME-hard, CONP-complete) [25]. But the complexity of the reachability problem for flat FIFO systems was not known. Only the complexity of the control-state reachability problem was known to be NP-complete for flat FIFO systems [21]. Moreover, other properties and model-checking have not been studied for flat FIFO systems.

Contributions. We solve the open problem of the complexity of the reachability problem for flat FIFO systems by showing that it is NP-complete; we extend this result to other usual verification properties and show that they are also NP-complete. Then we show that a flat FIFO system can be simulated by a synchronized product of counter systems. This synchronized product is flattable and its reachability set is semilinear.

2 Preliminaries

We write $\mathbb{Z}$ (resp. $\mathbb{N}$) to denote the set of integers (resp. non-negative integers). A finite alphabet is any finite set Σ. Its elements are referred to as letters; Σ^* is the set of all finite sequences of letters, referred to as words. We denote by $w_1 w_2$ the word obtained by concatenating w_1 and w_2; and ϵ is the empty sequence, which is the unity for the concatenation operation. We write Σ^+ for $\Sigma^* \setminus \{\epsilon\}$. If w_1 is a prefix of w_2, we denote by $w_1^{-1} w_2$ the word obtained from w_2 by dropping the prefix w_1. If w_1 is not a prefix of w_2, then $w_1^{-1} w_2$ is undefined. A word $z \in \Sigma^*$ is primitive if $z \notin w^* \setminus \{w\}$ for any $w \in \Sigma^*$. We



(a) Process P. **(b)** Process Q. **(c)** Process R.

Figure 1 FIFO system of Example 2.2.

denote by $Parikh(w) : \Sigma \to \mathbb{N}$ the function that maps each letter $a \in \Sigma$ to the number of times a occurs in w. We denote by w^n the concatenation of n copies of w. The infinite word x^ω is obtained by concatenating x infinitely many times.

FIFO Systems.

▶ **Definition 2.1** (FIFO systems). *A FIFO system S is a tuple (Q, F, M, Δ) where Q is a finite set of control states, F is a finite set of FIFO channels, M is a finite message alphabet and $\Delta \subseteq (Q \times Q) \cup (Q \times (F \times \{!, ?\} \times M) \times Q)$ is a finite set of transitions.*

We write a transition $(q, (\mathsf{c}, ?, a), q')$ as $q \xrightarrow{\mathsf{c}?a} q'$; we similarly modify other transitions. We call q the source state and q' the target state. Transitions of the form $q \xrightarrow{\mathsf{c}?a} q'$ (resp. $q \xrightarrow{\mathsf{c}!a} q'$) denote retrieve actions (resp. send actions). Transitions of the form $q \longrightarrow q'$ do not change the channel contents but only change the control state.

The channels in F hold strings in M^*. Given two channel valuations $\mathbf{w}_1, \mathbf{w}_2 \in (M^*)^F$, we denote by $\mathbf{w}_1 \cdot \mathbf{w}_2$ the valuation obtained by concatenating the contents in $\mathbf{w}_1$ and $\mathbf{w}_2$ channel-wise. For a letter $a \in M$ and a channel $\mathsf{c} \in F$, we denote by $\mathbf{a}_\mathsf{c}$ the channel valuation that assigns a to c and ϵ to all other channels. The semantics of a FIFO system S is given by a transition system T_S whose set of states is $Q \times (M^*)^F$, also called configurations. Every transition $q \xrightarrow{\mathsf{c}?a} q'$ of S and channel valuation $\mathbf{w} \in (M^*)^F$ results in the transition $(q, \mathbf{a}_\mathsf{c} \cdot \mathbf{w}) \xrightarrow{\mathsf{c}?a} (q', \mathbf{w})$ in T_S. Every transition $q \xrightarrow{\mathsf{c}!a} q'$ of S and channel valuation $\mathbf{w} \in (M^*)^F$ results in the transition $(q, \mathbf{w}) \xrightarrow{\mathsf{c}!a} (q', \mathbf{w} \cdot \mathbf{a}_\mathsf{c})$ in T_S. Intuitively, the transition $q \xrightarrow{\mathsf{c}?a} q'$ (resp. $q \xrightarrow{\mathsf{c}!a} q'$) retrieves the letter a from the front of the channel c (resp. sends the letter a to the back of the channel c). A run of S is a (finite or infinite) sequence of configurations $(q_0, \mathbf{w}_0)(q_1, \mathbf{w}_1) \cdots$ such that for every $i \geq 0$, there is a transition t_i such that $(q_i, \mathbf{w}_i) \xrightarrow{t_i} (q_{i+1}, \mathbf{w}_{i+1})$.

▶ **Example 2.2.** Let us present a (distributed) FIFO system (from [30]) with three processes P, Q, R that communicate through four FIFO channels pq, qp, pr, rq. Processes are extended finite automata where transitions are labeled by sending or receiving operations with FIFO channels and, for example, channel pq is an unidirectional FIFO channel from process P to process Q. From this distributed FIFO system, we get a FIFO system as given in Definition 2.1 by product construction. The control states of the product FIFO system are triples, containing control states of processes P, Q, R. The product FIFO system can go from one control state to another if one of the processes goes from a control state to another and the other two processes remain in their states. For example, the product system has the transition $(q_1, q_2, q_3) \xrightarrow{\mathsf{pq}!a_1} (q_1', q_2, q_3)$, if process P has the transition $q_1 \xrightarrow{\mathsf{pq}!a_1} q_1'$.

(a) Flat FIFO system. **(b)** Path schema denoted by $p_0(\ell_1)^*p_1(l_2)^*p_2$.

Figure 2 Example flat FIFO system and path schema.

For analyzing the running time of algorithms, we assume the size of a system to be the number of bits needed to specify a system (and source/target configurations if necessary) using a reasonable encoding. Let us begin to present the reachability problems that we tackle in this paper.

▶ Problem (Reachability). *Given:* A FIFO system S and two configurations $(q_0, \mathbf{w}_0)$ and $(q, \mathbf{w})$. *Question:* Is there a run starting from $(q_0, \mathbf{w}_0)$ and ending at $(q, \mathbf{w})$?

▶ Problem (Control-state reachability). *Given:* A FIFO system S, a configuration $(q_0, \mathbf{w}_0)$ and a control-state q. *Question:* Is there a channel valuation $\mathbf{w}$ such that $(q, \mathbf{w})$ is reachable from $(q_0, \mathbf{w}_0)$?

It is folklore that reachability and control-state reachability are undecidable for machines operating on FIFO channels.

Flat systems. For a FIFO system $S = (Q, F, M, \Delta)$, its *system graph* G_S is a directed graph whose set of vertices is Q. There is a directed edge from q to q' if there is some transition $q \xrightarrow{\mathsf{c}?a} q'$ or $q \xrightarrow{\mathsf{c}!a} q'$ for some channel c and some letter a, or there is a transition $q \longrightarrow q'$. We say that S is *flat* if in G_S, every vertex is in at most one directed cycle. Figure 2a shows a flat FIFO system.

We call a FIFO system $S = (Q, F, M, \Delta)$ a *path segment* from state q_0 to state q_r if $Q = \{q_0, \ldots, q_r\}$, $\Delta = \{t_1, \ldots, t_r\}$ and for every $i \in \{1, \ldots, r\}$, q_{i-1} is the source of t_i and q_i is its target. We call a FIFO system $S = (Q, F, M, \Delta)$ an *elementary loop* on q_0 if $Q = \{q_0, \ldots, q_r\}$, $\Delta = \{t_1, \ldots, t_{r+1}\}$ and for each $i \in \{1, \ldots, r+1\}$, t_i has source q_{i-1} and target $q_{i \mod (r+1)}$. We call $t_1 \cdots t_{r+1}$ the label of the loop. A *path schema* is a flat FIFO system comprising of a sequence $p_0\ell_1p_1\ell_2p_2 \cdots l_rp_r$, where $p_0, \ldots, p_r$ are path segments and $\ell_1, \ldots, \ell_r$ are elementary loops. There are states $q_0, q_1, \ldots, q_{r+1}$ such that p_0 is a path segment from q_0 to q_1 and for every $i \in \{1, \ldots, r\}$, p_i is a path segment from q_i to q_{i+1} and ℓ_i is an elementary loop on q_i. Except q_i, none of the other states in ℓ_i appear in other path segments or elementary loops. To emphasize that $\ell_1, \ldots, \ell_r$ are elementary loops, we denote the path schema as $p_0(\ell_1)^*p_1 \cdots (\ell_r)^*p_r$. We use the term elementary loop to distinguish them from loops in general, which may have some states appearing more than once. All loops in flat FIFO systems are elementary. Figure 2b shows a path schema, where wavy lines indicate long path segments or elementary loops that may have many intermediate states and transitions. This path schema is obtained from the flat FIFO system of Figure 2a by removing the transitions from q_1 to q_3, q_4 to q_5 and q_6 to q_3.

▶ Remark 2.3 (Fig. 1). Each process P, Q, R is flat and the cartesian product of the three automata is almost flat except on one state: there are two loops, one sending y in channel pq and another one retrieving y from channel pq.

Notations and definitions. For any sequence σ of transitions of a FIFO system and channel $\mathsf{c} \in F$, we denote by y^σ_c (resp. x^σ_c) the sequence of letters sent to (resp. retrieved from) the channel c by σ. For a configuration $(q, \mathbf{w})$, let $\mathbf{w}(\mathsf{c})$ denote the contents of channel c.

Equations on words. We recall some classical results reasoning about words and prove of one of them, to be used later. Proofs of this and a few other results are omitted. All the proofs can be found in the full version of this paper, which is on HAL with the same title. The well-known Levi's Lemma says that the words $u, v \in \Sigma^*$ that are solutions of the equation $uv = vu$ satisfy $u, v \in z^*$ where z is a primitive word. The solutions of the equation $uv = vw$ satisfy $u = xy, w = yx, v = (xy)^n x$, for some words x, y and some integer $n \geq 0$. The following lemma is used in [25] for exactly the same purpose as here.

▶ **Lemma 2.4.** *Consider three finite words $x, y \in \Sigma^+$ and $w \in \Sigma^*$. The equation $x^\omega = wy^\omega$ holds iff there exists a primitive word $z \neq \epsilon$ and two words x', x'' such that $x = x'x''$, $x''x' \in z^*$, $w \in x^*x'$ and $y \in z^*$.*

3 Complexity of Reachability Properties for Flat FIFO Systems

In this section, we give complexity bounds for the reachability problem for flat FIFO systems. We also establish the complexity of other related problems, viz. repeated control state reachability, termination, boundedness, channel boundedness and letter channel boundedness. We use the algorithm for repeated control state reachability as a subroutine for solving termination and boundedness. For channel boundedness and letter channel boundedness, we use another argument based on integer linear programming.

In [21], Esparza, Ganty, and Majumdar studied the complexity of reachability for highly undecidable models (multipushdown systems) but synchronized by bounded languages in the context of bounded model-checking. In particular, they proved that control-state reachability is NP-complete for flat FIFO systems (in fact for FIFO systems controlled by a bounded language). The NP upper bound is based on a simulation of FIFO path schemas by pushdown systems. Some constraints need to be imposed on the pushdown systems to ensure the correctness of the simulation. The structure of path schemas enables these constraints to be expressed as linear constraints on integer variables and this leads to the NP upper bound.

Surprisingly, the NP upper bound in [21] is given only for the control-state reachability problem; the complexity of the reachability problem is not established in [21] while it is given for all other considered models. However, there is a simple linear reduction from reachability to control-state reachability for FIFO (and Last In First Out) systems [32]. Such reductions are not known to exist for other models like counter systems and vector addition systems.

We begin by reducing reachability to control-state reachability (personal communication from Grégoire Sutre [32]) for (general and flat) FIFO systems.

▶ **Proposition 3.1** ([32]). *Reachability reduces (with a linear reduction) to control-state reachability, for general FIFO systems and for flat FIFO systems.*

▶ Remark 3.2. Control-state reachability is reducible to reachability for general FIFO systems. Suppose $\Sigma = \{a_1, \ldots, a_d\}$ and there are $\mathtt{p}$ channels. Using the same notations as in the previous proof, from A and q, one constructs the system $B_{A,q}$ as follows: one adds, to A, $d \times \mathtt{p}$ self loops $\ell_{i,j}$, each labeled by $\mathtt{j}?a_i$, for $i \in \{1, .., d\}$ and $j \in \{1, \ldots, \mathtt{p}\}$, all from and to the control-state q. We infer that q is reachable in A if and only if (by definition) there exists $\mathbf{w}$ such that $(q, \mathbf{w})$ is reachable in A if and only if (q, ϵ) is reachable in $B_{A,q}$. Here, (q, ϵ) denotes the configuration where q is the control state and all channels are empty. Note that $B_{A,q}$ is not necessarily flat, even if A is flat.

It is proved in [21, Theorem 7] that control state reachability is in NP for flat FIFO systems. Combining this with Proposition 3.1, we immediately deduce:

▶ **Corollary 3.3.** *Reachability is in* NP *for flat FIFO systems.*

Now we define problems concerned with infinite behaviors.

▶ Problem (Repeated reachability). *Given:* A FIFO system S, two configurations $(q_0, \mathbf{w}_0)$ and $(q, \mathbf{w})$. *Question:* Is there an infinite run from $(q_0, \mathbf{w}_0)$ such that $(q, \mathbf{w})$ occurs infinitely often along this run?

▶ Problem (Cyclicity). *Given:* A FIFO system S and a configuration $(q, \mathbf{w})$. *Question:* Is $(q, \mathbf{w})$ reachable (by a non-empty run) from $(q, \mathbf{w})$?

▶ Problem (Repeated control-state reachability). *Given:* A FIFO system S, a configuration $(q_0, \mathbf{w}_0)$ and a control-state q. *Question:* Is there an infinite run from $(q_0, \mathbf{w}_0)$ such that q occurs infinitely often along this run?

We can easily obtain an NP upper bound for repeated reachability in flat FIFO systems. A non-deterministic Turing machine first uses the previous algorithm for reachability (Corollary 3.3) to verify that $(q, \mathbf{w})$ is reachable from $(q_0, \mathbf{w}_0)$. Then the same algorithm is used again to verify that $(q, \mathbf{w})$ is reachable from $(q, \mathbf{w})$ (i.e. cyclic).

▶ **Corollary 3.4.** *Repeated reachability is in* NP *for flat FIFO systems.*

Let us recall that the cyclicity property is EXPSPACE-complete for Petri nets [8, 20] while structural cyclicity (every configuration is cyclic) is in PTIME. Let us show that one may decide the cyclicity property for flat FIFO systems in linear time.

▶ **Lemma 3.5.** *In a flat FIFO system, a configuration $(q, \mathbf{w})$ is reachable from $(q, \mathbf{w})$ iff there is an elementary loop labeled by σ, such that $(q, \mathbf{w}) \xrightarrow{\sigma} (q, \mathbf{w})$.*

To decide whether $(q, \mathbf{w}) \xrightarrow{*} (q, \mathbf{w})$, one tests whether $(q, \mathbf{w}) \xrightarrow{\sigma} (q, \mathbf{w})$ for some elementary loop σ in the flat FIFO system. Since the FIFO system is flat, q can be in at most one loop, so only one loop need to be tested. This gives a linear time algorithm for deciding cyclicity.

▶ **Corollary 3.6.** *Testing cyclicity can be done in linear time for flat FIFO systems.*

We are now going to show an NP upper bound for repeated control state reachability.

Let a loop be labeled with σ. Recall that for each channel c, we denote by x_{c}^{σ} (resp. y_{c}^{σ}) the projection of σ to letters retrieved from (resp. sent to) the channel c. Let us write σ_{c} for the projection of σ on channel c.

▶ Remark 3.7. The loop labeled by σ is infinitely iterable from $(q, \mathbf{w})$ iff σ_{c} is infinitely iterable from $(q, \mathbf{w}(\mathsf{c}))$, for every channel c. If σ is infinitely iterable from $(q, \mathbf{w})$ then each projection σ_{c} is also infinitely iterable from $(q, \mathbf{w}(\mathsf{c}))$. Conversely, suppose σ_{c} is infinitely iterable from $(q, \mathbf{w}(\mathsf{c}))$, for every channel c. For all $\mathsf{c} \neq \mathsf{c}'$, the actions of σ_{c} and $\sigma_{\mathsf{c}'}$ are on different channels and hence independent of each other. Since σ is a shuffle of $\{\sigma_{\mathsf{c}} \mid \mathsf{c} \in F\}$, we deduce that σ is infinitely iterable from $(q, \mathbf{w})$.

We now give a characterization for a loop to be infinitely iterable.

▶ **Lemma 3.8.** *Suppose an elementary loop is on a control state q and is labeled by σ. It is infinitely iterable starting from the configuration $(q, \mathbf{w})$ iff for every channel c, $x_{\mathsf{c}}^{\sigma} = \epsilon$ or the following three conditions are true: σ is fireable at least once from $(q, \mathbf{w})$, $(x_{\mathsf{c}}^{\sigma})^{\omega} = \mathbf{w}(\mathsf{c}) \cdot (y_{\mathsf{c}}^{\sigma})^{\omega}$ and $|x_{\mathsf{c}}^{\sigma}| \leq |y_{\mathsf{c}}^{\sigma}|$.*

Proof. Let ℓ be an elementary loop on a control state q and labeled by σ. If σ is infinitely iterable starting from the configuration $(q, \mathbf{w})$ then for every channel c, one has $|x_\mathsf{c}| \leq |y_\mathsf{c}|$. Otherwise, $|x_\mathsf{c}| > |y_\mathsf{c}|$ (the number of letters retrieved is more than the number of letters sent in each iteration), so the size of the channel content reduces with each iteration, so there is a bound on the number of possible iterations. Since σ is infinitely iterable from $(q, \mathbf{w})$, the inequation $(x_\mathsf{c}^\sigma)^n \leq \mathbf{w}(\mathsf{c}) \cdot (y_\mathsf{c}^\sigma)^n$ must hold for all $n \geq 0$ (here, $\leq$ denotes the prefix relation). If $x_\mathsf{c} \neq \epsilon$, we may go at the limit and we obtain $(x_\mathsf{c}^\sigma)^\omega \leq \mathbf{w}(\mathsf{c}) \cdot (y_\mathsf{c}^\sigma)^\omega$.

Finally, σ is fireable at least once from $(q, \mathbf{w})$ since it is fireable infinitely from $(q, \mathbf{w})$.

Now conversely, suppose that for every channel c, $x_\mathsf{c}^\sigma = \epsilon$ or the following three conditions are true: σ is fireable at least once from $(q, \mathbf{w})$, $(x_\mathsf{c}^\sigma)^\omega = \mathbf{w}(\mathsf{c}) \cdot (y_\mathsf{c}^\sigma)^\omega$ and $|x_\mathsf{c}^\sigma| \leq |y_\mathsf{c}^\sigma|$. For the rest of this proof, we fix a channel c and write $x_\mathsf{c}^\sigma, y_\mathsf{c}^\sigma, \mathbf{w}(\mathsf{c})$ as x, y, w to simplify the notation.

If $x = \epsilon$ then σ is infinitely iterable because it doesn't retrieve anything. So assume that $x \neq \epsilon$. We have $x^\omega = wy^\omega$ from the hypothesis. We infer from Lemma 2.4 that there is a primitive word $z \neq \epsilon$ and words x', x'' such that $x = x'x''$, $x''x' \in z^*$, $w \in x^*x'$ and $y \in z^*$. Suppose $x''x' = z^j$ and $y = z^k$. Since $|y| \geq |x| = |x''x'|$, we have $k \geq j$. Let us prove the following monotonicity property: for all $n \geq 0$, σ is fireable from any channel content wz^n and the resulting channel content is $wz^{n+(k-j)}$ (this will imply that for all $m \geq 1$, $w \xrightarrow{\sigma^m} wz^{m\times(k-j)}$, hence that σ is infinitely iterable). We prove the monotonicity property by induction on n.

For the base case $n = 0$, we need to prove that $w \xrightarrow{\sigma} wz^{k-j}$. By hypothesis, σ is fireable at least once from w, hence $w \xrightarrow{\sigma} w'$ for some w'. We have $w' = x^{-1}wy = x^{-1}x^r x' z^k$ for some $r \in \mathbb{N}$. Since $k \geq j$, we have $w' = x^{-1}x^r x' z^j z^{k-j} = x^{-1}x^r x'(x''x')z^{k-j} = x^{-1}x^r(x'x'')x'z^{k-j} = x^{-1}x^{r+1}x'z^{k-j} = x^r x' z^{k-j} = wz^{k-j}$.

For the induction step, we have to show that σ is fireable from channel content wz^{n+1} and the resulting channel content is $wz^{n+1+(k-j)}$. From induction hypothesis, we know that σ is fireable from channel content wz^n. Since $y = z^k$, the channel content after firing a prefix σ_1 of σ is $x_1^{-1}wz^n z^s z_1$, where x_1 is some prefix of x, $s \in \mathbb{N}$ and z_1 is some prefix of z. By induction on $|\sigma_1|$, we can verify that σ_1 can be fired from wz^{n+1} and results in $x_1^{-1}wz^{n+1}z^s z_1$. Hence, σ can be fired from wz^{n+1} and results in $x^{-1}wz^{n+1}y = x^{-1}x^r x' z^{n+1} z^k = x^{-1}x^r x' z^j z^{n+1+k-j} = x^{-1}x^r x' x'' x' z^{n+1+k-j} = x^{-1}x^{r+1}x'z^{n+1+k-j} = wz^{n+1+k-j}$. This completes the induction step and hence proves the monotonicity property.

Hence σ is infinitely iterable. ◀

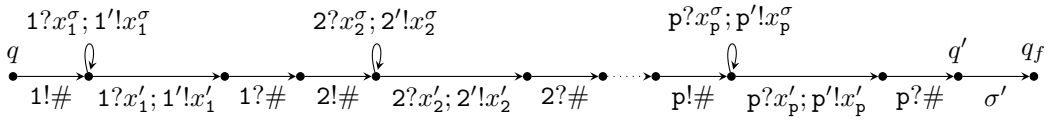
The proof of Lemma 3.8 provides a complete characterization of the contents of a FIFO channel when a loop is infinitely iterable. One may observe that the channel acts like a counter (of the number of occurrences of z).

▶ **Corollary 3.9.** *With the previous notations, the set of words in channel* c *that occur in control-state* q *is the regular periodic language* $\mathbf{w}(\mathsf{c}) \cdot [z_\mathsf{c}^{k-j}]^*$*, when the elementary loop containing* q *is iterated arbitrarily many times.*

▶ Remark 3.10. One may find other similar results on infinitely iterable loops in many papers [22, 29, 6, 7, 25]. Our Lemma 3.8 is the same as [25, Proposition 5.1] except that it (easily) extends it to systems with *multiple* channels and also provides the converse. Lemma 3.8 simplifies and improves Proposition 5.4. in [7] that used the equivalent but more complex notion of *inc-repeating sequence*. Also, the results in [7] don't give the simple representation of the regular periodic language.

▶ **Lemma 3.11.** *The repeated control state reachability problem is in* NP *for flat FIFO systems.*

Proof. We describe an NP algorithm. Suppose S is the given flat FIFO system and the control state q is to be reached repeatedly. Suppose q is in a loop labeled with σ. The algorithm first verifies that for every channel $\mathtt{c}$, $|x_{\mathtt{c}}^{\sigma}| \leq |y_{\mathtt{c}}^{\sigma}|$ – if this condition is violated, the answer is no. From Lemma 3.8, it is enough to verify that we can reach a configuration $(q, \mathbf{w})$ such that σ can be fired at least once from $(q, \mathbf{w})$ and for every channel $\mathtt{c}$ for which $x_{\mathtt{c}}^{\sigma} \neq \epsilon$, we have $(x_{\mathtt{c}}^{\sigma})^{\omega} = \mathbf{w}(\mathtt{c}) \cdot (y_{\mathtt{c}}^{\sigma})^{\omega}$. Since the case of $x_{\mathtt{c}}^{\sigma} = \epsilon$ can be handled easily, we assume in the rest of this proof that $x_{\mathtt{c}}^{\sigma} \neq \epsilon$ for every $\mathtt{c}$. For verifying that $(x_{\mathtt{c}}^{\sigma})^{\omega} = \mathbf{w}(\mathtt{c}) \cdot (y_{\mathtt{c}}^{\sigma})^{\omega}$, the algorithm depends on Lemma 2.4: the algorithm guesses $x'_{\mathtt{c}}, x''_{\mathtt{c}}, z_{\mathtt{c}} \in M^*$ such that $x_{\mathtt{c}}^{\sigma} = x'_{\mathtt{c}} x''_{\mathtt{c}}$ and $x''_{\mathtt{c}} x'_{\mathtt{c}}, y_{\mathtt{c}}^{\sigma} \in z_{\mathtt{c}}^*$. We have $|x'_{\mathtt{c}}|, |x''_{\mathtt{c}}| \leq |x_{\mathtt{c}}^{\sigma}|$ and $|z_{\mathtt{c}}| \leq |y_{\mathtt{c}}^{\sigma}|$ so the guessed strings are of size bounded by the size of the input. It remains to verify that we can reach a configuration $(q, \mathbf{w})$ such that for every channel $\mathtt{c}$, $\mathbf{w}(\mathtt{c}) \in (x_{\mathtt{c}}^{\sigma})^* x'_{\mathtt{c}}$ and σ can be fired at least once from $(q, \mathbf{w})$. For accomplishing these two tasks, we add a channel $\mathtt{c}'$ for every channel $\mathtt{c}$ in the FIFO system S. The following gadgets are appended to the control state q, assuming that there are $\mathtt{p}$ channels and $\#$ is a special letter not in the channel alphabet M. We denote by σ' the sequence of transitions obtained from σ by replacing every channel $\mathtt{c}$ by $\mathtt{c}'$. A transition labeled with $\mathtt{c}?x_{\mathtt{c}}^{\sigma}; \mathtt{c}'!x_{\mathtt{c}}^{\sigma}$ is to be understood as a sequence of transitions whose effect is to retrieve $x_{\mathtt{c}}^{\sigma}$ from channel $\mathtt{c}$ and send $x_{\mathtt{c}}^{\sigma}$ to channel $\mathtt{c}'$.



Finally our algorithm runs the NP algorithm to check that the control state q_f is reachable. We claim that the control state q can be visited infinitely often iff our algorithm accepts. Suppose q can be visited infinitely often. So the loop containing q can be iterated infinitely often. Hence from Lemma 3.8, we infer that S can reach a configuration $(q, \mathbf{w})$ such that σ can be fired at least once and for every channel $\mathtt{c}$, $|x_{\mathtt{c}}^{\sigma}| \leq |y_{\mathtt{c}}^{\sigma}|$ and $(x_{\mathtt{c}}^{\sigma})^{\omega} = \mathbf{w}(\mathtt{c}) \cdot (y_{\mathtt{c}}^{\sigma})^{\omega}$. From Lemma 2.4, there exist $x'_{\mathtt{c}}, x''_{\mathtt{c}}, z_{\mathtt{c}} \in M^*$ such that $x_{\mathtt{c}}^{\sigma} = x'_{\mathtt{c}} x''_{\mathtt{c}}$, $\mathbf{w}(\mathtt{c}) \in (x_{\mathtt{c}}^{\sigma})^* x'_{\mathtt{c}}$ and $x''_{\mathtt{c}} x'_{\mathtt{c}}, y_{\mathtt{c}}^{\sigma} \in z_{\mathtt{c}}^*$. Our algorithm can guess exactly these words $x'_{\mathtt{c}}, x''_{\mathtt{c}}, z_{\mathtt{c}}$. It is easy to verify that from the configuration $(q, \mathbf{w})$, the configuration $(q', \mathbf{w}')$ can be reached, where $\mathbf{w}'(\mathtt{c}') = \mathbf{w}(\mathtt{c})$ for every $\mathtt{c}$. Since σ can be fired from $(q, \mathbf{w})$, σ' can be fired from $(q', \mathbf{w}')$ to reach q_f. So our algorithm accepts.

Conversely, suppose our algorithm accepts. Hence the control state q_f is reachable. By construction, we can verify that the run reaching the control state q_f has to visit a configuration $(q, \mathbf{w})$ such that for every channel $\mathtt{c}$, $\mathbf{w}(\mathtt{c}) \in (x_{\mathtt{c}}^{\sigma})^* x'_{\mathtt{c}}$ and σ can be fired at least once from $(q, \mathbf{w})$. Our algorithm also verifies that $|x_{\mathtt{c}}^{\sigma}| \leq |y_{\mathtt{c}}^{\sigma}|$, $x_{\mathtt{c}}^{\sigma} = x'_{\mathtt{c}} x''_{\mathtt{c}}$ and $x''_{\mathtt{c}} x'_{\mathtt{c}}, y_{\mathtt{c}}^{\sigma} \in z_{\mathtt{c}}^*$. Hence, from Lemma 2.4 and Lemma 3.8, we infer that the loop containing q can be iterated infinitely often starting from the configuration $(q, \mathbf{w})$. Hence, there is a run that visits q infinitely often. ◀

Let us now introduce the non-termination and the unboundedness problems.

▶ Problem (Non-termination). *Given:* A FIFO system S and an initial configuration $(q_0, \mathbf{w}_0)$. *Question:* Is there an infinite run from $(q_0, \mathbf{w}_0)$?

▶ Problem (Unboundedness). *Given:* A FIFO system S and an initial configuration $(q_0, \mathbf{w}_0)$. *Question:* Is the set of configurations reachable from $(q_0, \mathbf{w}_0)$ infinite?

▶ **Corollary 3.12.** *For flat FIFO systems, the non-termination and unboundedness problems are in* NP.

For a word w and a letter a, $|w|_a$ denotes the number of occurrences of a in w. For a FIFO system, we say that a letter a is unbounded in channel c if for every number B, there exists a reachable configuration $(q, \mathbf{w})$ with $|\mathbf{w}(\mathsf{c})|_a \geq B$. A channel c is unbounded if at least one letter a is unbounded in c.

▶ Problem (Channel-unboundedness). *Given:* A FIFO system S, an initial configuration $(q_0, \mathbf{w}_0)$ and a channel c. *Question:* Is the channel c unbounded from $(q_0, \mathbf{w}_0)$?

▶ Problem (Letter-channel-unboundedness). *Given:* A FIFO system S, an initial configuration $(q_0, \mathbf{w}_0)$, a channel c and a letter a. *Question:* Is the letter a unbounded in channel c from $(q_0, \mathbf{w}_0)$?

Now we give an NP upper bound for letter channel unboundedness in flat FIFO systems. We use the following two results in our proof.

▶ **Theorem 3.13** ([21, Theorem 3, Theorem 7]). *Let $S = p_0(\ell_1)^* p_1 \cdots (\ell_r)^* p_r$ be a FIFO path schema. We can compute in polynomial time an existential Presburger formula $\phi(x_1, \ldots, x_r)$ satisfying the following property: there is a run of S in which the loop ℓ_i is iterated exactly n_i times for every $i \in \{1, \ldots, r\}$ iff $\phi(n_1, \ldots, n_r)$ is true.*

For vectors $\mathbf{k}, \mathbf{x}$ and matrix $\mathbf{A}$, the expression $\mathbf{k} \cdot \mathbf{x}$ denotes the dot product and the expression $\mathbf{A}\mathbf{x}$ denotes the matrix product.

▶ **Theorem 3.14** ([31, Lemma 3]). *Suppose $\mathbf{A}$ is an integer matrix and $\mathbf{k}, \mathbf{b}$ are integer vectors satisfying the following property: for every $B \in \mathbb{N}$, there exists a vector $\mathbf{x}$ of* rational numbers *such that $\mathbf{A}\mathbf{x} \geq \mathbf{b}$ and $\mathbf{k} \cdot \mathbf{x} \geq B$. If there is an integer vector $\mathbf{x}$ such that $\mathbf{A}\mathbf{x} \geq \mathbf{b}$, then for every $B \in \mathbb{N}$, there exists an* integer *vector $\mathbf{x}$ such that $\mathbf{A}\mathbf{x} \geq \mathbf{b}$ and $\mathbf{k} \cdot \mathbf{x} \geq B$.*

▶ **Theorem 3.15.** *Given a flat FIFO system, a letter a and channel c, the problem of checking whether a is unbounded in c is in* NP.

Proof. The letter a is unbounded in c iff there exists a control state q such that for every number B, there is a reachable configuration with control state q and at least B occurrences of a in channel c (this follows from definitions since there are only finitely many control states). A non-deterministic polynomial time Turing machine begins by guessing a control state q. If there are r loops in the path schema ending at q, the Turing machine computes an existential Presburger formula $\phi(x_1, \ldots, x_r)$ satisfying the following property: $\phi(n_1, \ldots, n_r)$ is true iff there is a run ending at q in which loop i is iterated n_i times for every $i \in \{1, \ldots, r\}$. Such a formula can be computed in polynomial time (Theorem 3.13). Let k_i be the number of occurrences of the letter a sent to channel c by one iteration of the i^{th} loop (k_i would be negative if a is retrieved instead). If loop i is iterated n_i times for every i in a run, then at the end of the run there are $k_1 n_1 + \cdots + k_r n_r$ occurrences of the letter a in channel c. To check that a is unbounded in channel c, we have to verify that there are tuples $\langle n_1, \ldots, n_r \rangle$ such that $\phi(n_1, \ldots, n_r)$ is true and $k_1 n_1 + \cdots + k_r n_r$ is arbitrarily large. This is easier to do if there are no disjunctions in the formula $\phi(x_1, \ldots, x_r)$. If there are any sub-formulas with disjunctions, the Turing machine non-deterministically chooses one of the disjuncts and drops the other one. This is continued till all disjuncts are discarded. This results in a conjunction of linear inequalities, say $A\mathbf{x} \geq \mathbf{b}$, where $\mathbf{x}$ is the tuple of variables $\langle x_1, \ldots, x_r \rangle$. The machine then tries to maximize $k_1 x_1 + \cdots + k_r x_r$ over rationals subject to the constraints $A\mathbf{x} \geq \mathbf{b}$. This can be done in polynomial time, since linear programming is in polynomial time. If the value $k_1 x_1 + \cdots + k_r x_r$ is unbounded above over rationals subject to the constraints $A\mathbf{x} \geq \mathbf{b}$, then the machine invokes the NP algorithm to check if the constraints $A\mathbf{x} \geq \mathbf{b}$ has a feasible solution over integers. If it does, then $k_1 x_1 + \cdots + k_r x_r$ is also unbounded above over integers (Theorem 3.14). Hence, in this case, a is unbounded in channel c. ◀

The above result also gives an NP upper bound for channel-unboundedness. We just guess a letter a and check that it is unbounded in the given channel.

We adapt the proof of NP-hardness for the control state reachability problem from [21] to prove NP hardness for reachability, repeated control state reachability, unboundedness and non-termination.

▶ **Lemma 3.16.** *For flat FIFO systems, reachability, repeated control-state reachability, non-termination, unboundedness, channel-unboundedness and letter-channel-unboundedness are NP-hard.*

Hence we deduce the main result of this Section.

▶ **Theorem 3.17** (Most properties are NP-complete)**.** *For flat FIFO systems, reachability, repeated reachability, repeated control-state reachability, termination, boundedness, channel-boundedness and letter-channel-boundedness are NP-complete. Cyclicity can be decided in linear time.*

4 Construction of an Equivalent Counter System

Suppose we want to model check flat FIFO systems against logics in which atomic formulas are of the form $\#^a_{\mathsf{c}} \geq k$, which means there are at least k occurrences of the letter a in channel c. There is no easy way of designing an algorithm for this model checking problem based on the construction in [21], even though we solved reachability and related problems in previous sections using that construction. That construction is based on simulating FIFO systems using automata that have multiple reading heads on an input tape. The channel contents of the FIFO system are represented in the automaton as the sequence of letters on the tape between two reading heads. There is no way in the automaton to access the tape contents between two heads, and hence no way to check the number of occurrences of a specific letter in a channel. CQDDs introduced in [7] represent the entire set of reachable states and they are also not suitable for model checking. To overcome this problem, we introduce here a counter system to simulate flat FIFO systems. This has the additional advantage of being amenable to analysis using existing tools on counter machines.

Counter systems are finite state automata augmented with counters that can store natural numbers. Let K be a finite set of counters and let *guards over* K be the set $G(K)$ of positive Boolean combinations[1] of constraints of the form $C = 0$ and $C > 0$, where $C \in K$.

▶ **Definition 4.1** (Counter systems)**.** *A counter system S is a tuple $\langle Q, K, \Delta \rangle$ where Q is a finite set of control states and $\Delta \subseteq Q \times G(K) \times \{-1, 0, 1\}^K \times Q$ is a finite set of transitions.*

We may add one or two labeling functions to the tuple $\langle Q, K, \Delta \rangle$ to denote labeled counter systems. The semantics of a counter system is a transition system with set of states $Q \times \mathbb{N}^K$, called *configurations of the counter system*. A counter valuation $\nu \in \mathbb{N}^K$ *satisfies* a guard $C = 0$ (resp. $C > 0$) if $\nu(C) = 0$ (resp. $\nu(C) > 0$), written as $\nu \models C = 0$ (resp. $\nu \models C > 0$). The satisfaction relation is extended to Boolean combinations in the standard way. For every transition $\delta = q \xrightarrow[g]{\mathbf{u}} q'$ in the counter system, we have transitions $(q, \nu_1) \xrightarrow{\delta} (q', \nu_2)$ in the associated transition system for every ν_1 such that $\nu_1 \models g$ and $\nu_2 = \nu_1 + \mathbf{u}$ (addition of

[1] In the literature, counter systems can have more complicated guards, such as Presburger constraints. For our purposes, this restricted version suffices.

vectors is done component-wise). We write a transition $(q, C_2 = 0, \langle 1, 0\rangle, q')$ as $q \xrightarrow[C_2=0]{C_1^{++}} q'$, denoting addition of 1 to C_1 by C_1^{++}. We denote by $\longrightarrow$ the union $\cup_{\delta\in\Delta} \xrightarrow{\delta}$. A *run* of the counter system is a finite or infinite sequence $(q_0, \nu_0) \longrightarrow (q_1, \nu_1) \longrightarrow \cdots$ of configurations, where each pair of consecutive configurations is in the transition relation.

We assume for convenience that the message alphabet M of a FIFO system is the disjoint union of $M_1, \ldots, M_{\mathtt{p}}$, where $M_{\mathtt{c}}$ is the alphabet for channel $\mathtt{c}$. In the following, let $S = (Q, F, M, \Delta)$ be a flat FIFO system, where the set of channels $F = \{1, \ldots, \mathtt{p}\}$ and the set of transitions $\Delta = \{t_1, \ldots, t_r\}$.

The *counting abstraction system* corresponding to S is a labeled counter system $S_{\text{count}} = (Q, K, \Delta_{\text{count}}, \psi, T)$, where $(Q, K, \Delta_{\text{count}})$ is a counter system and ψ, T are labeling functions. The set of *counters* K is in bijection with $M \times \Delta$ and a counter will be denoted $c_{a,t}$ or shortly (a, t), for $a \in M$ and $t \in \Delta$. The set Δ_{count} of *transitions* of S_{count} and the labeling functions $\psi : \Delta_{\text{count}} \to (M \times \Delta) \cup \{\tau\}$ and $T : \Delta_{\text{count}} \to \Delta$ are defined as follows: for every transition $t \in \Delta$, one adds the following transitions in Δ_{count} :

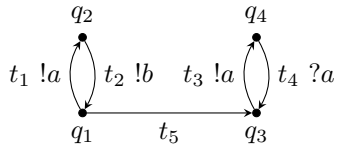
- If t *sends* a message, $t = q_1 \xrightarrow{\mathtt{c}!a} q_2$, then the transition $t_{\text{count}} = q_1 \xrightarrow{(a,t)^{++}} q_2$ is added to Δ_{count} ; we define $\psi(t_{\text{count}}) = \tau$ and $T(t_{\text{count}}) = t$.
- If $t = q_1 \longrightarrow q_2$ doesn't change any channel content, then the transition $t_{\text{count}} = q_1 \longrightarrow q_2$ is added to Δ_{count} ; we define $\psi(t_{\text{count}}) = \tau$ and $T(t_{\text{count}}) = t$.
- If t *receives* a message, $t = q_1 \xrightarrow{\mathtt{c}?a} q_2$, then the set of transitions A_t is added to Δ_{count} with $A_t = \{\delta_{a,t'} = q_1 \xrightarrow[(a,t')>0]{(a,t')^{--}} q_2 \mid t' \text{ sends } a \text{ to channel } \mathtt{c}\}$. We define $\psi(\delta_{a,t'}) = (a, t')$ and $T(\delta_{a,t'}) = t$, for all $\delta_{a,t'} \in A_t$.

The function ψ above will be used for synchronization with other counter systems later and T will be used to match the traces of this counter system with those of the original flat FIFO system. In figures, we do not show the labels given by ψ and T. They can be easily determined. For a transition $\delta_{a,t'} \in \Delta_{\text{count}}$, it decrements the counter (a, t') and $\psi(\delta_{a,t'}) = (a, t')$. Transitions that don't decrement any counter are mapped to τ by ψ.

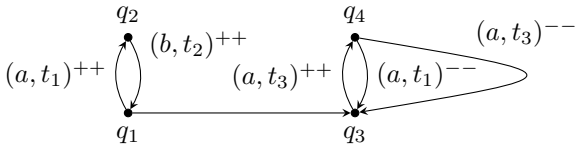
▶ **Example 4.2.** Figure 3a shows a flat FIFO system and Fig. 3b shows its counting abstraction system.

The idea behind the counting abstraction system is to *ignore the order of letters* stored in the channels and use counters to remember only the number of occurrences of each letter. If a transition t sends letter a, the corresponding transition in the counting abstraction system increments the counter (a, t). If a transition t retrieves a letter a, the retrieved letter would have been produced by some earlier transition t'; the corresponding transition in the counting abstraction system will decrement the counter (a, t'). The counting abstraction system doesn't exactly simulate the flat FIFO system. For example, if the transition labeled $(a, t_1)^{--}$ in Fig. 3b is executed, we know that there is at least one occurrence of the letter a in the channel, since the counter (a, t_1) is greater than zero at the beginning of the transition. However, it is not clear that the letter a is at the front of the channel; there might be an occurrence of the letter b at the front. This condition can't be tested using the counting abstraction system. We use other counter systems to maintain the order of letters.

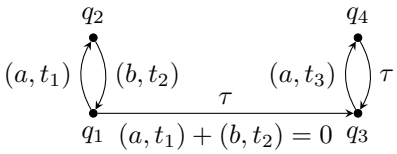
The *order system for channel* $\mathtt{c}$ is a labeled counter system $S^{\mathtt{c}}_{\text{order}} = (Q, K, \Delta^{\mathtt{c}}_{\text{order}}, \psi^{\mathtt{c}})$, where $(Q, K, \Delta^{\mathtt{c}}_{\text{order}})$ is a counter system and $\psi^{\mathtt{c}}$ is a labeling function. The set of control states Q and the set of counters K are the same as in the counting abstraction system. The set $\Delta^{\mathtt{c}}_{\text{order}}$ of *transitions* of $S^{\mathtt{c}}_{\text{order}}$ and the *labeling function* $\psi^{\mathtt{c}} : \Delta^{\mathtt{c}}_{\text{order}} \to (M \times \Delta) \cup \{\tau\}$ are defined as follows: for every $t \in \Delta$, one adds the following transitions in $\Delta^{\mathtt{c}}_{\text{order}}$:



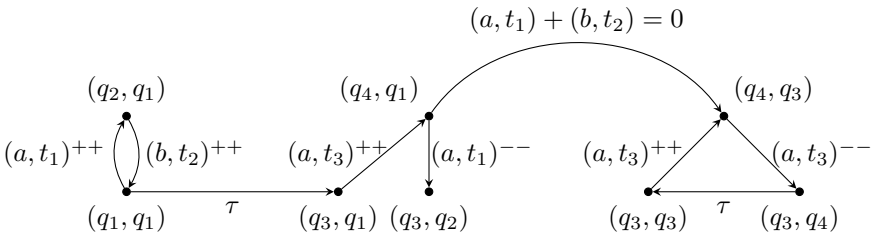
(a) Flat FIFO system.



(b) Counting abstraction system.



(c) Order system.



(d) Synchronized counter system.

Figure 3 An example flat FIFO system and the equivalent counter system.

- If $t = q_1 \xrightarrow{\mathsf{c}!a} q_2$, one adds to $\Delta^{\mathsf{c}}_{\text{order}}$ the transition $t' = q_1 \to q_2$ and $\psi^{\mathsf{c}}(t') = (a, t)$.
- If $t = q_1 \xrightarrow{x} q_2$ where x doesn't contain a sending operation (of a letter) to channel c, one adds to $\Delta^{\mathsf{c}}_{\text{order}}$ the transition $t' = q_1 \to q_2$ and $\psi^{\mathsf{c}}(t') = \tau$.

While adding the transitions above, if t happens to be the first transition after and outside a loop in S, we add a guard to the transition t' that we have given in the above two cases. Suppose t is the first transition after and outside a loop, and the loop is labeled by σ. We add the following guard to the transition t'.

$$\sum_{\substack{t'' \text{ occurs in } \sigma \\ a \in M}} (a, t'') = 0$$

Figure 3c shows the order system corresponding to the flat FIFO system of Fig. 3a.

We will synchronize the counting abstraction system with the order systems by rendez-vous on transition labels. Suppose the order system is in state q_2 as shown in Fig. 3c. The only transition going out from q_2 is labeled by (b, t_2), denoting the fact that the front of the channel contains b. The counting abstraction system can't execute the transition labeled with $(a, t_1)^{--}$ in this configuration, since its ψ-label is (a, t_1) and hence it can't synchronize with the order system, whose next transition is labeled with (b, t_2). The guard $(a, t_1)+(b, t_2) = 0$ in the bottom transition in Fig. 3c ensures that all occurrences of letters produced by iterations of the first loop are retrieved before those produced by the second loop.

In the following, the *label of a transition* refers to the image of that transition under the function ψ (if the transition is in the counting abstraction system) or the function ψ^{c} (if the transition is in the order system for channel c). The *synchronized counter system* $S_{\text{sync}} = S_{\text{count}} \mid\mid S^1_{\text{order}} \mid\mid \ldots \mid\mid S^{\mathsf{c}}_{\text{order}} \mid\mid \ldots \mid\mid S^{\mathsf{p}}_{\text{order}}$ is the synchronized (by rendez-vous)

product of the *counting abstraction system* S_{count} and the *order systems* $S^{\mathtt{c}}_{\text{order}}$ for all channels $\mathtt{c} \in \{1, \ldots, \mathtt{p}\}$. All counter systems share the same set of counters K and have disjoint copies of the set of control states Q, so the global control states of the synchronized counter system are tuples in $Q^{\mathtt{p}+1}$. Transitions labeled with τ need not synchronize with others. Each transition labeled (by the function ψ or $\psi^{\mathtt{c}}$ as explained above) with an element of $M \times \Delta$ should synchronize with exactly one other transition that is similarly labeled. We extend the labeling function T of S_{count} to S_{sync} as follows: if a transition t of S_{count} participates in a transition t_s of S_{sync}, then $T(t_s) = T(t)$. If no transition from S_{count} participates in t_s, then $T(t_s) = \tau$ and we call t_s a silent transition.

Since we have assumed that the channel alphabets for different channels are mutually disjoint, synchronizations can only happen between the counting abstraction system and one of the order systems. For a global control state $\overline{q} \in Q^{\mathtt{p}+1}$, $\overline{q}(0)$ denotes the local state of the counting abstraction system and $\overline{q}(\mathtt{c})$ denotes the local state of the order system for channel $\mathtt{c}$. The synchronized counter system maintains the channel contents of the flat FIFO system as explained next.

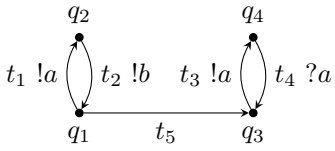
We now explain that every reachable configuration $(\overline{q}, \nu)$ of S_{sync} corresponds to a unique configuration $h(\overline{q}, \nu)$ of the original FIFO system S. The corresponding configuration of S is $(\overline{q}(0), h_1(v_1), h_2(v_2), \ldots h_{\mathtt{p}}(v_{\mathtt{p}}))$, where the words $v_{\mathtt{c}} \in \Delta^*$ and morphisms $h_{\mathtt{c}} : \Delta^* \to M^*$ are as follows. Fix a channel $\mathtt{c}$. Let $v_{\mathtt{c}} \in \Delta^*$ be a word labelling a path in S from $\overline{q}(\mathtt{c})$ to $\overline{q}(0)$ such that $\mathit{Parikh}(v_{\mathtt{c}})(t) = \nu((a,t))$ for every transition $t \in \Delta$ that sends some letter to channel $\mathtt{c}$ (and a is the letter that is sent by t). Now, define $h_{\mathtt{c}}(t) = a$ if t sends some letter to channel $\mathtt{c}$ (and a is the letter sent) and $h_{\mathtt{c}}(t) = \epsilon$ otherwise. The word $h_c(v_{\mathtt{c}})$ is unique since S is flat and so the set of traces of S, interpreted as a language over the alphabet Δ, is included in a bounded language. Intuitively, the path $v_{\mathtt{c}}$ gives the order of letters in channel $\mathtt{c}$ and the counters give the number of occurrences of each letter.

▶ **Example 4.3.** Figure 3d shows the reachable states of the synchronized counter system for the flat FIFO system in Fig. 3a. Initially, both the counting abstraction system and the order system are in state q_1, so the global state is (q_1, q_1). Then the counting abstraction system may execute the transition labeled $(a, t_1)^{++}$ and go to state q_2 while the order system stays in state q_1, resulting in the global state (q_2, q_1). Consider the global state $\overline{q} = (q_3, q_2)$ and counter valuation ν with $\nu((a,t_1)) = 2, \nu((b,t_2)) = 3$ and $\nu((a,t_3)) = 1$. Then, for the only channel $\mathtt{c} = 1$, $v_{\mathtt{c}} = t_2(t_1t_2)^2t_5t_3$ and $h_{\mathtt{c}}(v_{\mathtt{c}}) = b(ab)^2a$.

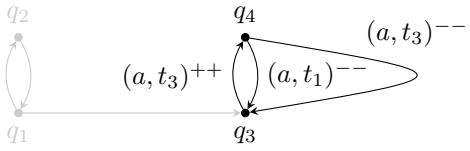
A relation R between the reachable configurations of the FIFO system S and the synchronized counter system S_{sync} is a *weak bisimulation* if every pair $((q, \mathbf{w}), (\overline{q}, \nu)) \in R$ satisfies the following conditions: (1) for every transition $(q, \mathbf{w}) \xrightarrow{t} (q', \mathbf{w}')$ in S, there is a sequence σ of transitions in S_{sync} such that $T(\sigma) \in \tau^* t \tau^*$, $(\overline{q}, \nu) \xrightarrow{\sigma} (\overline{q'}, \nu')$ and $((q', \mathbf{w}'), (\overline{q'}, \nu')) \in R$, (2) for every transition $(\overline{q}, \nu) \xrightarrow{t_s} (\overline{q'}, \nu')$ in S_{sync} with $T(t_s) = \tau$, $((q, \mathbf{w}), (\overline{q'}, \nu')) \in R$ and (3) for every transition $(\overline{q}, \nu) \xrightarrow{t_s} (\overline{q'}, \nu')$ in S_{sync} with $T(t_s) = t \neq \tau$, $(q, \mathbf{w}) \xrightarrow{t} (q', \mathbf{w}')$ is a transition in S and $((q', \mathbf{w}'), (\overline{q'}, \nu')) \in R$.

▶ **Lemma 4.4.** *The relation* $\{(h((\overline{q}, \nu)), (\overline{q}, \nu)) \mid (\overline{q}, \nu)$ *is reachable in* $S_{\text{sync}}\}$ *is a weak bisimulation.*

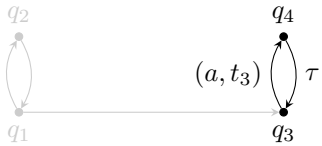
The synchronized counter system S_{sync} is not flat. E.g., there are two transitions from q_4 to q_3 in Fig. 3b. Those two states are in more than one loop, violating the condition of flatness. However, suppose a run is visiting states q_3, q_4 of the counting abstraction system and states q_3, q_4 of the order system as shown in Fig. 4 (parts of the systems that are no longer reachable are greyed out). Now the transition labeled $(a, t_1)^{--}$ can't be used and the



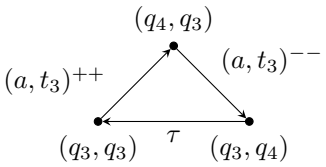
(a) Flat FIFO system.



(b) Counting abstraction system (grey part no longer reachable).



(c) Order system (grey part no longer reachable).



(d) Part of synchronized counter system still reachable.

Figure 4 Flattening.

run is as shown in Fig. 4d, which is a flat counter system. In general, suppose $\ell_0, \ell_1, \ldots, \ell_r$ are the loops in S. There is a flat counter system S_{flat} whose set of runs is the set of runs ρ of the synchronized transition system which satisfy the following property: in ρ, all local states of the counting abstraction system are in some loop ℓ_i and for every channel c, all local states of the order system $S^{\mathsf{c}}_{\text{order}}$ are in some loop ℓ_{c}. This is the intuition for the next result.

Let traces(S_{sync}) be the set of all runs of S_{sync}. Let S' be another counter system with set of states Q' and the same set of counters as S_{sync} and let $f : Q' \to Q$ be a function. We say that S' is a f-flattening of S_{sync} [15, Definition 6] if S' is flat and for every transition $q \xrightarrow[g]{u} q'$ of S', $f(q) \xrightarrow[g]{u} f(q')$ is a transition in S_{sync}. Further, S' is a f-trace-flattening of S_{sync} [15, Definition 8] if S' is a f-flattening of S_{sync} and traces(S_{sync}) $= f(\text{traces}(S'))$.

▶ **Lemma 4.5.** *The synchronized counter system S_{sync} is trace-flattable.*

Let S_{flat} be a trace-flattening of S_{sync}. In general, the size of S_{flat} is exponential in the size of S_{sync}, which is exponential in the size of S. The weak bisimulation shown in Lemma 4.4 can be strengthened to bisimulation; see the full version for details. In theory, problems on flat FIFO systems can be solved by using tools on counter systems (bisimulation preserves CTL* and trace-flattening preserves LTL [15, Theorem1]). It remains to be seen if tools can be optimized to make verifying FIFO systems work in practice.

5 Conclusion and Perspectives

We answered the complexity of the main reachability problems for flat FIFO systems which are NP-complete as for flat counter systems. We also show how to translate a flat FIFO system into a trace-flattable counter system. This opens the way to model-check general FIFO systems by *enumerating their flat subsystems*. For example, if we construct the product of the three processes shown in Fig. 1, the resulting FIFO system is not flat. It does become flat if we remove the self loop labeled $\mathtt{pq}?y$. The resulting flat subsystem is unbounded, so it implies that the original system is also unbounded. Hence, even if the given FIFO system is not flat, some questions can often be answered by analyzing flat subsystems. This strategy has worked well for counter systems and offers hope for FIFO systems.

References

1 Parosh Aziz Abdulla, Aurore Collomb-Annichini, Ahmed Bouajjani, and Bengt Jonsson. Using Forward Reachability Analysis for Verification of Lossy Channel Systems. *Formal Methods in System Design*, 25(1):39–65, 2004. doi:10.1023/B:FORM.0000033962.51898.1a.

2 Aurore Annichini, Ahmed Bouajjani, and Mihaela Sighireanu. TReX: A Tool for Reachability Analysis of Complex Systems. In Gérard Berry, Hubert Comon, and Alain Finkel, editors, *Computer Aided Verification*, pages 368–372, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg.

3 Sébastien Bardin, Alain Finkel, Jérôme Leroux, and Laure Petrucci. FAST: Fast Acceleration of Symbolic Transition systems. In Warren A. Hunt, Jr and Fabio Somenzi, editors, *Proceedings of the 15th International Conference on Computer Aided Verification (CAV'03)*, volume 2725 of *Lecture Notes in Computer Science*, pages 118–121, Boulder, Colorado, USA, July 2003. Springer. URL: http://www.lsv.ens-cachan.fr/Publis/PAPERS/PS/FAST-cav03.ps.

4 Sébastien Bardin, Alain Finkel, Jérôme Leroux, and Philippe Schnoebelen. Flat acceleration in symbolic model checking. In Doron A. Peled and Yih-Kuen Tsay, editors, *Proceedings of the 3rd International Symposium on Automated Technology for Verification and Analysis (ATVA'05)*, volume 3707 of *Lecture Notes in Computer Science*, pages 474–488, Taipei, Taiwan, October 2005. Springer. doi:10.1007/11562948_35.

5 Bernard Boigelot. Domain-specific regular acceleration. *STTT*, 14(2):193–206, 2012. doi:10.1007/s10009-011-0206-x.

6 Bernard Boigelot, Patrice Godefroid, Bernard Willems, and Pierre Wolper. The Power of QDDs (Extended Abstract). In Pascal Van Hentenryck, editor, *Static Analysis, 4th International Symposium, SAS '97, Paris, France, September 8-10, 1997, Proceedings*, volume 1302 of *Lecture Notes in Computer Science*, pages 172–186. Springer, 1997. doi:10.1007/BFb0032741.

7 Ahmed Bouajjani and Peter Habermehl. Symbolic Reachability Analysis of FIFO-Channel Systems with Nonregular Sets of Configurations. *Theor. Comput. Sci.*, 221(1-2):211–250, 1999. doi:10.1016/S0304-3975(99)00033-X.

8 Zakaria Bouziane and Alain Finkel. Cyclic Petri Net Reachability Sets are Semi-Linear Effectively Constructible. In Faron Moller, editor, *Proceedings of the 2nd International Workshop on Verification of Infinite State Systems (INFINITY'97)*, volume 9 of *Electronic Notes in Theoretical Computer Science*, pages 15–24, Bologna, Italy, July 1997. Elsevier Science Publishers. URL: http://www.lsv.ens-cachan.fr/Publis/PAPERS/PDF/BF-infinity97.pdf.

9 Marius Bozga, Radu Iosif, and Filip Konecný. Safety Problems are NP-complete for Flat Integer Programs with Octagonal Loops. *CoRR*, abs/1307.5321, 2013. arXiv:1307.5321.

10 Marius Bozga, Radu Iosif, Filip Konecný, and Tomás Vojnar. Tool Demonstration of the FLATA Counter Automata Toolset. In Andrei Voronkov, Laura Kovács, and Nikolaj Bjørner, editors, *Second International Workshop on Invariant Generation, WING 2009, York, UK, March 29, 2009 and Third International Workshop on Invariant Generation, WING 2010, Edinburgh, UK, July 21, 2010*, volume 1 of *EPiC Series in Computing*, page 75. EasyChair, 2010. URL: http://www.easychair.org/publications/paper/51875.

11 Daniel Brand and Pitro Zafiropulo. On Communicating Finite-State Machines. *J. ACM*, 30(2):323–342, 1983. doi:10.1145/322374.322380.

12 Nadia Busi, Roberto Gorrieri, Claudio Guidi, Roberto Lucchi, and Gianluigi Zavattaro. Choreography and Orchestration Conformance for System Design. In Paolo Ciancarini and Herbert Wiklicky, editors, *Coordination Models and Languages, 8th International Conference, COORDINATION 2006, Bologna, Italy, June 14-16, 2006, Proceedings*, volume 4038 of *Lecture Notes in Computer Science*, pages 63–81. Springer, 2006. doi:10.1007/11767954_5.

13 Gérard Cécé and Alain Finkel. Verification of Programs with Half-Duplex Communication. *Information and Computation*, 202(2):166–190, November 2005. doi:10.1016/j.ic.2005.05.006.

14 Normann Decker, Peter Habermehl, Martin Leucker, Arnaud Sangnier, and Daniel Thoma. Model-checking Counting Temporal Logics on Flat Structures. In *28th International Conference*

on Concurrency Theory, CONCUR 2017, LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2017.

15 S. Demri, A. Finkel, V. Goranko, and G. van Drimmelen. Towards a Model-Checker for Counter Systems. In Susanne Graf and Wenhui Zhang, editors, *Automated Technology for Verification and Analysis*, pages 493–507, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

16 Stéphane Demri, Amit Dhar, and Arnaud Sangnier. Equivalence Between Model-Checking Flat Counter Systems and Presburger Arithmetic. *Theoretical Computer Science*, 2017. Special issue of RP'14, to appear.

17 Stéphane Demri, Amit Kumar Dhar, and Arnaud Sangnier. On the Complexity of Verifying Regular Properties on Flat Counter Systems. In Fedor V. Fomin, Rūsiņš Freivalds, Marta Kwiatkowska, and David Peleg, editors, *Proceedings of the 40th International Colloquium on Automata, Languages and Programming (ICALP'13) – Part II*, volume 7966 of *Lecture Notes in Computer Science*, pages 162–173, Riga, Latvia, July 2013. Springer. doi:10.1007/978-3-642-39212-2_17.

18 Stéphane Demri, Amit Kumar Dhar, and Arnaud Sangnier. Taming past LTL and flat counter systems. *Inf. Comput.*, 242:306–339, 2015. doi:10.1016/j.ic.2015.03.007.

19 Stéphane Demri, Alain Finkel, Valentin Goranko, and Govert van Drimmelen. Model-checking CTL* over Flat Presburger Counter Systems. *Journal of Applied Non-Classical Logics*, 20(4):313–344, 2010. doi:10.3166/jancl.20.313-344.

20 Frank Drewes and Jérôme Leroux. Structurally Cyclic Petri Nets. *Logical Methods in Computer Science*, 11(4), 2015. doi:10.2168/LMCS-11(4:15)2015.

21 Javier Esparza, Pierre Ganty, and Rupak Majumdar. A Perfect Model for Bounded Verification. In *Proceedings of the 2012 27th Annual IEEE/ACM Symposium on Logic in Computer Science*, LICS '12, pages 285–294, Washington, DC, USA, 2012. IEEE Computer Society. doi:10.1109/LICS.2012.39.

22 Alain Finkel. *Structuration des systèmes de transitions: applications au contrôle du parallélisme par files fifo, Thèse d'Etat.* PhD thesis, Université Paris-Sud, Orsay, 1986.

23 Alain Finkel and Jean Goubault-Larrecq. Forward Analysis for WSTS, Part II: Complete WSTS. *Logical Methods in Computer Science*, 8(3:28), September 2012. doi:10.2168/LMCS-8(3:28)2012.

24 Alain Finkel and Étienne Lozes. Synchronizability of Communicating Finite State Machines is not Decidable. In Ioannis Chatzigiannakis, Piotr Indyk, Anca Muscholl, and Fabian Kuhn, editors, *Proceedings of the 44th International Colloquium on Automata, Languages and Programming (ICALP'17)*, volume 80 of *Leibniz International Proceedings in Informatics*, pages 122:1–122:14, Warsaw, Poland, July 2017. Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.ICALP.2017.122.

25 Alain Finkel, S. Purushothaman Iyer, and Grégoire Sutre. Well-Abstracted Transition Systems: Application to FIFO Automata. *Information and Computation*, 181(1):1–31, February 2003. URL: http://www.lsv.ens-cachan.fr/Publis/PAPERS/PS/FPS-ICOMP.ps.

26 Blaise Genest, Dietrich Kuske, and Anca Muscholl. On Communicating Automata with Bounded Channels. *Fundam. Inform.*, 80(1-3):147–167, 2007. URL: http://content.iospress.com/articles/fundamenta-informaticae/fi80-1-3-09.

27 Christoph Haase. *On the complexity of model checking counter automata.* PhD thesis, University of Oxford, UK, 2012.

28 Radu Iosif and Arnaud Sangnier. How Hard is It to Verify Flat Affine Counter Systems with the Finite Monoid Property? In Cyrille Artho, Axel Legay, and Doron Peled, editors, *Automated Technology for Verification and Analysis - 14th International Symposium, ATVA 2016, Chiba, Japan, October 17-20, 2016, Proceedings*, volume 9938 of *Lecture Notes in Computer Science*, pages 89–105, 2016. doi:10.1007/978-3-319-46520-3_6.

29 Thierry Jéron and Claude Jard. Testing for Unboundedness of FIFO Channels. *Theor. Comput. Sci.*, 113(1):93–117, 1993. doi:10.1016/0304-3975(93)90212-C.

30 Julien Lange and Nobuko Yoshida. Verifying Asynchronous Interactions via Communicating Session Automata. *CoRR*, abs/1901.09606, 2019. `arXiv:1901.09606`.

31 Christos H. Papadimitriou. On the Complexity of Integer Programming. *J. ACM*, 28(4):765–768, October 1981. `doi:10.1145/322276.322287`.

32 Gregoire Sutre. Personal communication, 2018.

33 Gregor von Bochmann. Communication protocols and error recovery procedures. *Operating Systems Review*, 9(3):45–50, 1975.