Logical Methods in Computer Science
Volume 16, Issue 4, 2020, pp. 4:1–4:29
https://lmcs.episciences.org/

Submitted Dec. 25, 2019
Published Oct. 13, 2020

VERIFICATION OF FLAT FIFO MACHINES

ALAIN FINKEL [a,c] AND M. PRAVEEN [b,c]

[a] Université Paris-Saclay, ENS Paris-Saclay, CNRS, Laboratoire Spécification et Vérification, 91190, Gif-sur-Yvette, France and Institut Universitaire de France

[b] Chennai Mathematical Institute, India

[c] UMI ReLaX

Abstract. The decidability and complexity of reachability problems and model-checking for flat counter machines have been explored in detail. However, only few results are known for flat (lossy) FIFO machines, 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 machines, generalizing similar existing results for flat counter machines. We also show that reachability is NP-complete for flat lossy FIFO machines and for flat front-lossy FIFO machines. We construct a trace-flattable system of many counter machines communicating via rendez-vous that is bisimilar to a given flat FIFO machine, which allows to model-check the original flat FIFO machine. Our results lay the theoretical foundations and open the way to build a verification tool for (general) FIFO machines based on analysis of flat sub-machines.

1. Introduction

FIFO machines. Asynchronous distributed processes communicating through First In First Out (FIFO) channels are used since the seventies as models for protocols [38], distributed and concurrent programming and more recently for web service choreography interface [15]. Since FIFO machines simulate counter machines, most reachability properties are *undecidable* for FIFO machines: for example, the basic task of checking if the number of messages buffered in a channel can grow unboundedly is undecidable [14].

There aren't many interesting and useful FIFO subclasses with a *decidable* reachability problem. Considering FIFO machines with a unique FIFO channel is not a useful restriction

---

Key words and phrases: Infinite state machines, FIFO, counters, flat machines, reachability, termination, complexity.

A preliminary version of this work appeared in the conference CONCUR 2019. 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)..

Partially supported by a grant from the Infosys foundation.

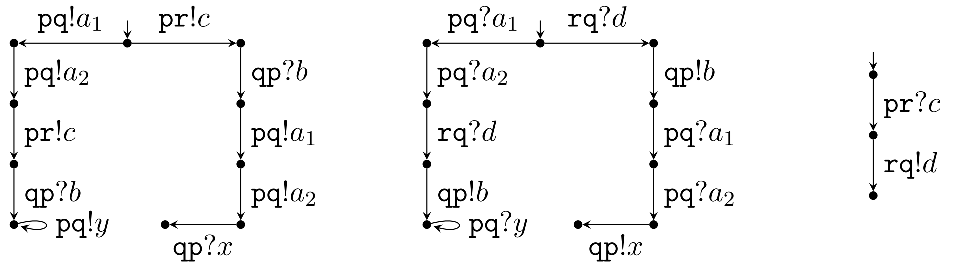
DOI:10.23638/LMCS-16(4:4)2020
© Alain Finkel [a,c] and M. Praveen [b,c]
Creative Commons

since they may simulate Turing machines [14]. A few examples of decidable subclasses are half-duplex systems [16] (but they are restricted to two machines since the natural extension to three machines leads to undecidability), existentially bounded deadlock free FIFO machines [29] (but it is undecidable to check if a machine is existentially bounded, even for deadlock free FIFO machines), synchronisable FIFO machines (the property of synchronisability is undecidable [27] and moreover, it is not clear which properties of synchronisable machines are decidable), flat FIFO machines [8, 10] and lossy FIFO machines [2] (but one loses the perfect FIFO mechanism).

Flat machine. A flat machine [6, 26, 17, 7] is a machine with a single 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 $w_1^* w_2^* \cdots w_k^*$ where every w_i is a non empty word. Analyzing flat machines 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 machines are particularly interesting since one may under-approximate any machine by its flat submachines.

For counter machines [22, 31], this strategy lead to some tools like FAST [4], LASH, TREX [3], FLATA [13] which enumerate all flat submachines 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 [4], 80% of the examples (including Petri nets and multi-threaded Java programs) could be effectively verified. The complexity of flat counter machines is well-known: reachability is Np-complete for variations of flat counter machines [30, 12, 21], model-checking first-order formulae and linear μ-calculus formulae is Pspace-complete while model-checking Büchi automata is Np-complete [20]; equivalence between model-checking flat counter machines and Presburger arithmetic is established in [19].

Flat FIFO machines. We know almost nothing about flat FIFO machines, even the complexity of reachability is not known. Boigelot et al. [8] used recognizable languages (QDD) for accelerating loops in a subclass of flat FIFO machines, where there are restrictions on the number of channels that a loop can operate on. Bouajjani and Habermehl [10] 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) [28]. But the complexity of the reachability problem for flat FIFO machines was not known. Only the complexity of the control-state reachability problem was known to be Np-complete for single-path flat FIFO machines [24]. Moreover, other properties and model-checking have not been studied for flat FIFO machines. Similarly, Abdulla et al.'s studied the verification of lossy FIFO machines by accelerating loops and representing them by a class of regular expressions called Simple Regular Expressions (SRE) [1, 2] and gave a polynomial (quadratic) algorithm for computing the reachability set $\sigma^*(L)$ of a loop labeled by σ from a SRE language L. But the complexity of the reachability problem for flat lossy FIFO machines was not known.



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

FIGURE 1. FIFO system of Example 2.2 (from [33])

Contributions. We solve the open problem of the complexity of the reachability problem for flat FIFO machines by showing that it is NP-complete; we extend this result to other usual verification properties and show that they are also NP-complete. We also show that the reachability problem is NP-complete for flat (front-)lossy FIFO machines. Then we show that a flat FIFO machine can be simulated by a synchronized product of counter machines. 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_1w_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 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 Machines.

Definition 2.1 (FIFO machines)**.** A FIFO machine 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 $(q, (\mathtt{c}, ?, a), q')$ as $q \xrightarrow{\mathtt{c}?a} q'$; we similarly modify other transitions. We call q the source state and q' the target state. Transitions of the form $q \xrightarrow{\mathtt{c}?a} q'$ (resp. $q \xrightarrow{\mathtt{c}!a} q'$) 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^*. A channel valuation $\mathbf{w}$ is a fuction from F to M^*. We denote the set of all channel valuations by $(M^*)^F$. 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 $\mathtt{c} \in F$, we denote by $\mathbf{a}_\mathtt{c}$ the channel valuation that assigns a to c and ϵ to all other channels. The semantics of a FIFO machine 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. Figure 1 shows a FIFO system (from [33]) with three processes P, Q, R that communicate through four FIFO channels pq, qp, pr, rq. Processes are FIFO machines 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 FIFO system, we get a FIFO machine as given in Definition 2.1 by product construction.[1] The control states of the product FIFO machines are triples, containing control states of processes P, Q, R. The product FIFO machine 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 machine 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 $q_1 \xrightarrow{\mathsf{pq}!a_1} q_1'$.

For analyzing the running time of algorithms, we assume the size of a machine to be the number of bits needed to specify a machine (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 machine 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 $(q, \mathbf{w})$?

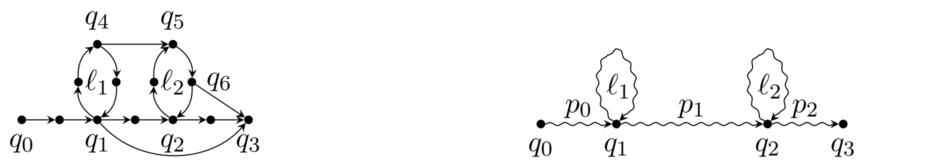
Problem (Control-state reachability). *Given:* A FIFO machine S, a configuration $(q_0, \mathbf{w}_0)$ and a control-state q. *Question:* Is there a channel valuation $\mathbf{w}$ such that $(q, \mathbf{w})$ is reachable from $(q_0, \mathbf{w}_0)$?

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

Flat machines. For a FIFO machine $S = (Q, F, M, \Delta)$, its *machine 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 2(a) shows a flat FIFO machine.

We call a FIFO machine $S = (Q, F, M, \Delta)$ 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 machine $S = (Q, F, M, \Delta)$ 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

[1]We use *FIFO machine* for one automaton and *FIFO system* when there are multiple automata interacting with each other.



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

Figure 2. Example flat FIFO machine and path schema

target $q_{i \bmod (r+1)}$. We call $t_1 \cdots t_{r+1}$ the label of the loop. A *path schema* is a flat FIFO machine comprising of a sequence $p_0\ell_1 p_1 \ell_2 p_2 \cdots l_r p_r$, 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 machines are elementary. Figure 2(b) 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 machine of Figure 2(a) 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 machine and channel $\mathsf{c} \in F$, we denote by y_{c}^{σ} (resp. x_{c}^{σ}) 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 one of them, to be used later. 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 [28] 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 $y \in z^*$.*

Proof. Suppose x, w, y satisfy the equation $x^\omega = w.y^\omega$. If $w = \epsilon$, then the equation reduces to $x^\omega = y^\omega$. Hence we deduce that $x^{|y|} = y^{|x|}$. In this case, we show (using Levi's Lemma and considering the three cases $\mid x \mid = \mid y \mid$ or $\mid x \mid < \mid y \mid$ or $\mid y \mid < \mid x \mid$) that the solutions are the words $x, y \in z^*$ where z is a finite primitive word. Now suppose that $w \neq \epsilon$, so choose the smallest $n \geq 0$ such that $w = x^n x'$ with $x = x'x''$. Hence, we obtain that $(x''x')^\omega = y^\omega$, and again we know that the solutions of this equation are $x''x', y \in z^*$ where z is a primitive word.

For the converse, suppose $x = x'x''$, $x''x' = z^j$, $w = x^n x'$ and $y = z^k$. We have $x^\omega = x^n x'(x''x')^\omega = w(z^j)^\omega = w(z^k)^\omega = wy^\omega$. □

3. Complexity of Reachability Properties for Flat FIFO Machines

In this section, we give complexity bounds for the reachability problem for flat FIFO machines. 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. Flat FIFO machines can simulate counter machines and reachability and related problems are known to be NP-hard for flat counter machines. However, the lower bound proofs for flat counter machines use binary encoding of counter updates, while the simulation of counter machines by FIFO machinbes use unary encoding. Hence, we cannot deduce lower bounds for flat FIFO machines from the lower bounds for flat counter machines. We prove the lower bounds for flat FIFO machines directly.

In [24], Esparza, Ganty, and Majumdar studied the complexity of reachability for highly undecidable models (multipushdown automata) 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 machines (in fact for single-path FIFO machines, i.e. FIFO machines controlled by a bounded language). The NP upper bound is based on a simulation of FIFO path schemas by multi head pushdown automata. Some constraints need to be imposed on the multi head pushdown automata 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 [24] is given only for the control-state reachability problem; the complexity of the reachability problem is not established in [24] 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) machines [37]. Such reductions are not known to exist for other models like counter machines and vector addition systems.

We begin by reducing reachability to control-state reachability (personal communication from Grégoire Sutre [37]) for (general and flat) FIFO machines.

Proposition 3.1 [37]**.** *Reachability reduces (with a linear reduction) to control-state reachability, for general FIFO machines and for flat FIFO machines.*

Proof. Let A be a FIFO machine, q a control-state and $(q, \mathbf{w})$ a configuration of A. We reduce reachability to control-state reachability. We construct the machine $B_{A,(q,\mathbf{w})}$ from A and $(q, \mathbf{w})$ as follows. The machine $B_{A,(q,\mathbf{w})}$ is obtained from A by adding a path to control state q as follows, where $\#$ is a new symbol not in M and $F = \{\mathtt{1}, \ldots, \mathtt{p}\}$. The transition labeled $\mathtt{1}?\mathbf{w}(\mathtt{1})\#$ is to be understood as a sequence of transitions whose effect is to retrieve the string $\mathbf{w}(\mathtt{1})\#$ from channel $\mathtt{1}$.

$$q \bullet \xrightarrow{\mathtt{1}!\#} \bullet \xrightarrow{\mathtt{1}?\mathbf{w}(\mathtt{1})\#} \bullet \cdots\cdots \bullet \xrightarrow{\mathtt{p}!\#} \bullet \xrightarrow{\mathtt{p}?\mathbf{w}(\mathtt{p})\#} \bullet\, q_{\text{stop}}$$

The configuration $(q, \mathbf{w})$ is reachable in A iff the control state q_{stop} is reachable in $B_{A,(q,\mathbf{w})}$. Note that if A is flat, then $B_{A,(q,\mathbf{w})}$ is also flat. □

Remark 3.2. Control-state reachability is reducible to reachability for general FIFO machines. 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 machine $B_{A,q}$ 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. Hence, this reduction does not imply NP-hardness of reachability in flat FIFO machines. We will prove NP-hardness later using a different reduction.

It is proved in [24, Theorem 7] that control state reachability is in NP for flat FIFO machine.

Corollary 3.3. *Reachability is in* NP *for flat FIFO machines.*

Now we define problems concerned with infinite behaviors.

Problem (Repeated reachability). *Given:* A FIFO machine 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 machine 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 machine 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 machines. 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 machines.*

Let us recall that the cyclicity property is EXPSPACE-complete for Petri nets [11, 23] while structural cyclicity (every configuration is cyclic) is in PTIME. Let us show that one may decide the cyclicity property for flat FIFO machines in linear time.

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

Proof. The implication from right to left ($\Leftarrow$) is clear. For the converse, suppose that $(q, \mathbf{w})$ is reachable from $(q, \mathbf{w})$. Flatness implies that q belongs to a (necessarily unique and elementary) loop, say a loop labeled by σ. As $(q, \mathbf{w})$ is reachable from $(q, \mathbf{w})$, there exists a sequence of transitions γ such that $(q, \mathbf{w}) \xrightarrow{\gamma} (q, \mathbf{w})$. Now, still from flatness, γ

is necessarily a power of σ, say $\gamma = \sigma^k$, $k \geq 1$. Hence we have: $(q, \mathbf{w}) \xrightarrow{\sigma^k} (q, \mathbf{w})$. Let us write $(q, \mathbf{w}) \xrightarrow{\sigma} (q, \mathbf{w}_1) \xrightarrow{\sigma} (q, \mathbf{w}_2) \xrightarrow{\sigma} \cdots \xrightarrow{\sigma} (q, \mathbf{w}_k) = (q, \mathbf{w})$. The effect of σ on the channel contents must preserve their initial length, so we have $\mid x^\sigma_{\mathsf{c}} \mid = \mid y^\sigma_{\mathsf{c}} \mid$ for every channel c. Since σ is fireable from $(q, \mathbf{w})$ and reaches $(q, \mathbf{w}_1)$, let us show that $\mathbf{w}_1 = \mathbf{w}$. If $x^\sigma_{\mathsf{c}} = \epsilon$ then $x^\sigma_{\mathsf{c}} = y^\sigma_{\mathsf{c}} = \epsilon$ and $\mathbf{w}_1 = \mathbf{w}$. So, let us suppose that $x^\sigma_{\mathsf{c}} \neq \epsilon$ (this also implies $y^\sigma_{\mathsf{c}} \neq \epsilon$). From $(q, \mathbf{w}) \xrightarrow{\sigma^k} (q, \mathbf{w})$, we know that the sequence σ^k is infinitely iterable and we have (1) $((x^\sigma_{\mathsf{c}})^k)^\omega = \mathbf{w}_{\mathsf{c}}((y^\sigma_{\mathsf{c}})^k)^\omega$ and since $k \geq 1$, $x^\sigma_{\mathsf{c}} \neq \epsilon$ and $y^\sigma_{\mathsf{c}} \neq \epsilon$, equality (1) implies that $(x^\sigma_{\mathsf{c}})^\omega = w(y^\sigma_{\mathsf{c}})^\omega$. In the rest of this proof, we skip the superscript σ and the subscript c for simplicity. We now write $x^\omega = wy^\omega$.

Lemma 2.4 implies that 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 $y \in z^*$. Let us write $y = z^d$. Since $x''x' \in z^*$ and since $x''x'$ has the same length as y, we deduce that $x''x' = z^d = y$. From $w \in x^*x'$, we obtain that $w \in (x'x'')^*x' = x'(x''x')^*$, hence $w \in x'(z^d)^*$. Hence, we have:

$$y = x''x' = z^d, x = x'x'' \text{ and } w = x'z^{ds} \text{ for some } s \geq 0 \quad (2)$$

Since $(q, \mathbf{w}) \xrightarrow{\sigma} (q, \mathbf{w}_1)$, the firing equation $w_1 = x^{-1}wy$ is satisfied. By replacing x, w by their values in (2) in the firing equation, we obtain:

$$w_1 = x^{-1}wy = x^{-1}x'z^{ds}z^d = x''^{-1}z^dz^{ds} = x''^{-1}x''x'z^{ds} = x'z^{ds} = w.$$

Hence $(q, \mathbf{w}) \xrightarrow{\sigma} (q, \mathbf{w})$. □

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 machine. Since the FIFO machine 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 machines.*

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^σ_{c} (resp. y^σ_{c}) 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, $x^\sigma_{\mathsf{c}} = \epsilon$ or the following three conditions are true: σ is fireable at least once from $(q, \mathbf{w})$, $(x^\sigma_{\mathsf{c}})^\omega = \mathbf{w}(\mathsf{c}) \cdot (y^\sigma_{\mathsf{c}})^\omega$ and $|x^\sigma_{\mathsf{c}}| \leq |y^\sigma_{\mathsf{c}}|$.*

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 $(x^\sigma_{\mathsf{c}})^n \leq \mathbf{w}(\mathsf{c}) \cdot (y^\sigma_{\mathsf{c}})^n$ 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 $(x^\sigma_{\mathsf{c}})^\omega \leq \mathbf{w}(\mathsf{c}) \cdot (y^\sigma_{\mathsf{c}})^\omega$.

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, $x^\sigma_{\mathsf{c}} = \epsilon$ or the following three conditions are true: σ is fireable at least once from $(q, \mathbf{w})$, $(x^\sigma_{\mathsf{c}})^\omega = \mathbf{w}(\mathsf{c}) \cdot (y^\sigma_{\mathsf{c}})^\omega$ and $|x^\sigma_{\mathsf{c}}| \leq |y^\sigma_{\mathsf{c}}|$.

For the rest of this proof, we fix a channel c and write $x^\sigma_{\mathsf{c}}, y^\sigma_{\mathsf{c}}, \mathbf{w}(\mathsf{c})$ 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 $w' = x^{-1}wy = x^{-1}x^rx'z^k$ for some $r \in \mathbb{N}$. Since $k \geq j$, we have $w' = x^{-1}x^rx'z^jz^{k-j} = x^{-1}x^rx'(x''x')z^{k-j} = x^{-1}x^r(x'x'')x'z^{k-j} = x^{-1}x^{r+1}x'z^{k-j} = x^rx'z^{k-j} = wz^{k-j}$.

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 $x_1^{-1}wz^nz^sz_1$, 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 $x_1^{-1}wz^{n+1}z^sz_1$. Hence, σ can be fired from wz^{n+1} and results in $x^{-1}wz^{n+1}y = x^{-1}x^rx'z^{n+1}z^k = x^{-1}x^rx'z^jz^{n+1+k-j} = x^{-1}x^rx'x''x'z^{n+1+k-j} = x^{-1}x^{r+1}x'z^{n+1+k-j} = wz^{n+1+k-j}$. This completes the induction step and hence proves the monotonicity property.

Hence σ is infinitely iterable. □

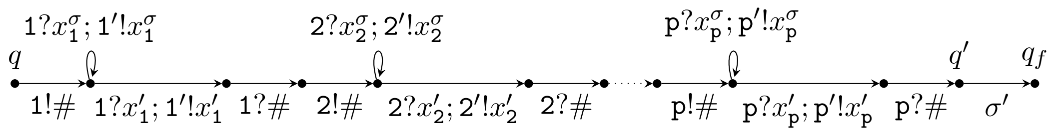
The proof of Lemma 3.8 provides a complete characterization of the contents of a FIFO channel when a loop is infinitely iterable. One may observe that the channel acts like a counter (of the number of occurrences of z).

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 [25, 32, 8, 10, 28]. Our Lemma 3.8 is the same as [28, Proposition 5.1] except that it (easily) extends it to machines with *multiple* channels and also provides the converse. Lemma 3.8 simplifies and improves Proposition 5.4 in [10] that used the equivalent but more complex notion of *inc-repeating sequence*. Also, the results in [10] don't give the simple representation of the regular periodic language.

Proposition 3.11. *The repeated control state reachability problem is in* NP *for flat FIFO machines.*

Proof. We describe an NP algorithm. Suppose S is the given flat FIFO machine 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 $\mathtt{c}$, $|x^\sigma_{\mathtt{c}}| \leq |y^\sigma_{\mathtt{c}}|$ — 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 $x^\sigma_{\mathtt{c}} \neq \epsilon$, we have $(x^\sigma_{\mathtt{c}})^\omega = \mathbf{w}(\mathtt{c}) \cdot (y^\sigma_{\mathtt{c}})^\omega$. Since the case of $x^\sigma_{\mathtt{c}} = \epsilon$ can be handled easily, we assume in the rest of this proof that $x^\sigma_{\mathtt{c}} \neq \epsilon$ for every $\mathtt{c}$. For verifying that $(x^\sigma_{\mathtt{c}})^\omega = \mathbf{w}(\mathtt{c}) \cdot (y^\sigma_{\mathtt{c}})^\omega$, the algorithm depends on Lemma 2.4: the algorithm guesses $x'_{\mathtt{c}}, x''_{\mathtt{c}}, z_{\mathtt{c}} \in M^*$ such that $x^\sigma_{\mathtt{c}} = x'_{\mathtt{c}} x''_{\mathtt{c}}$ and $x''_{\mathtt{c}} x'_{\mathtt{c}}, y^\sigma_{\mathtt{c}} \in z^*_{\mathtt{c}}$. We have $|x'_{\mathtt{c}}|, |x''_{\mathtt{c}}| \leq |x^\sigma_{\mathtt{c}}|$ and $|z_{\mathtt{c}}| \leq |y^\sigma_{\mathtt{c}}|$ 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 $\mathtt{c}$, $\mathbf{w}(\mathtt{c}) \in (x^\sigma_{\mathtt{c}})^* x'_{\mathtt{c}}$ 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 machine 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 $\mathtt{c}?x^\sigma_{\mathtt{c}}; \mathtt{c}'!x^\sigma_{\mathtt{c}}$ is to be understood as a sequence of transitions whose effect is to retrieve $x^\sigma_{\mathtt{c}}$ from channel $\mathtt{c}$ and send $x^\sigma_{\mathtt{c}}$ 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 $\mathtt{c}$, $|x^\sigma_{\mathtt{c}}| \leq |y^\sigma_{\mathtt{c}}|$ and $(x^\sigma_{\mathtt{c}})^\omega = \mathbf{w}(\mathtt{c}) \cdot (y^\sigma_{\mathtt{c}})^\omega$. From Lemma 2.4, there exist $x'_{\mathtt{c}}, x''_{\mathtt{c}}, z_{\mathtt{c}} \in M^*$ such that $x^\sigma_{\mathtt{c}} = x'_{\mathtt{c}} x''_{\mathtt{c}}$, $\mathbf{w}(\mathtt{c}) \in (x^\sigma_{\mathtt{c}})^* x'_{\mathtt{c}}$ and $x''_{\mathtt{c}} x'_{\mathtt{c}}, y^\sigma_{\mathtt{c}} \in z^*_{\mathtt{c}}$. 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 $\mathtt{c}$, $\mathbf{w}(\mathtt{c}) \in (x^\sigma_{\mathtt{c}})^* x'_{\mathtt{c}}$ and σ can be fired at least once from $(q, \mathbf{w})$. Our algorithm also verifies that $|x^\sigma_{\mathtt{c}}| \leq |y^\sigma_{\mathtt{c}}|$, $x^\sigma_{\mathtt{c}} = x'_{\mathtt{c}} x''_{\mathtt{c}}$ and $x''_{\mathtt{c}} x'_{\mathtt{c}}, y^\sigma_{\mathtt{c}} \in z^*_{\mathtt{c}}$. 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 machine 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 machine 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 machines, the non-termination and unboundedness problems are in* NP*.*

Proof. First we deal with non-termination. A flat machine is non-terminating iff there is an infinite run r. As there are only a finite number of control-states, the run will visit at least one control state (say q) infinitely often. Hence to solve non-termination, we can guess a control state q and use the NP algorithm of Proposition 3.11 to check that q can be visited infinitely often. This gives an NP upper bound for non-termination.

Next we deal with unboundedness. The *effect of a loop* ℓ labeled with σ is a vector of integers $\mathbf{v}_\ell \in \mathbb{Z}^F$ such that $\mathbf{v}_\ell(\mathsf{c}) = |x^\sigma_\mathsf{c}| - |y^\sigma_\mathsf{c}|$ for every $\mathsf{c} \in F$. If ℓ is an infinitely iterable loop, then $\mathbf{v}_\ell \geq \mathbf{0}$, where $\geq$ is component-wise comparison and $\mathbf{0}$ is the vector with all components equal to 0. If none of the loops in a flat FIFO machine are infinitely iterable, then only finitely many configurations can be reached. Hence, an unbounded flat FIFO machine has at least one loop ℓ that is infinitely iterable, hence $\mathbf{v}_\ell \geq \mathbf{0}$. If every infinitely iterable loop ℓ has $\mathbf{v}_\ell = \mathbf{0}$, then none of the infinitely iterable loops will increase the length of any channel content. Hence, there is a bound on the length of the channel contents in any reachable configuration, so only finitely many configurations can be reached. Hence, in an unbounded flat FIFO machine, there is at least one infinitely iterable loop ℓ with $\mathbf{v}_\ell \neq 0$.

Conversely, suppose a flat FIFO machine has an infinitely iterable loop ℓ with $\mathbf{v}_\ell \neq \mathbf{0}$. Since ℓ is infinitely iterable, $\mathbf{v}_\ell \geq \mathbf{0}$. Hence there is some channel c such that $\mathbf{v}_\ell(\mathsf{c}) \geq 1$. So every iteration of the loop ℓ will increase the length of the content of channel c by at least 1. Hence, infinitely many iterations of the loop ℓ will result in infinitely many configurations. So a machine S is unbounded iff there exists an infinitely iterable loop ℓ such that $\mathbf{v}_\ell \geq \mathbf{0}$ and $\mathbf{v}_\ell \neq \mathbf{0}$. Hence to decide unboundedness, we guess a control state q, verify that it belongs to a loop whose effect is non-negative on all channels and strictly positive on at least one channel and use the algorithm of Proposition 3.11 to check that q can be visited infinitely often. This gives an NP upper bound for unboundedness. □

For a word w and a letter a, $|w|_a$ denotes the number of occurrences of a in w. For a FIFO machine, 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 machine 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 machine 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 machines. We use the following two results in our proof.

Theorem 3.13 [24, 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.

Lemma 3.14 [34, 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* $\mathbf{k} \cdot \mathbf{x} \geq B$*.*

Proposition 3.15. *Given a flat FIFO machine, a letter* a *and channel* $\mathtt{c}$*, the problem of checking whether* a *is unbounded in* $\mathtt{c}$ *is in* NP*.*

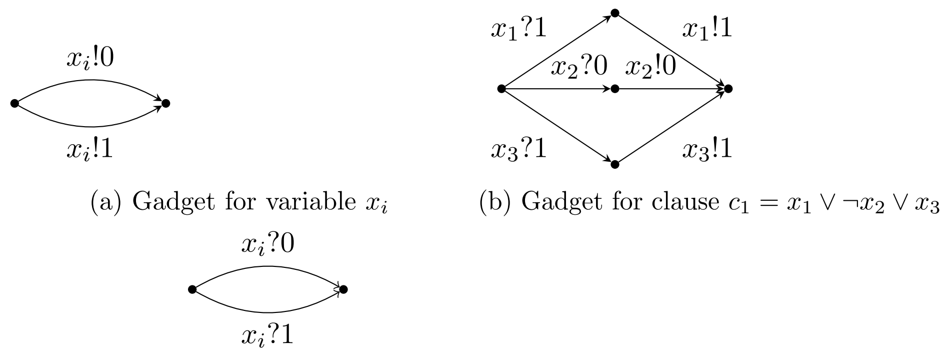
Proof. The letter a is unbounded in $\mathtt{c}$ iff there exists a control state q such that for every number B, there is a reachable configuration with control state q and at least B occurrences of a in channel $\mathtt{c}$ (this follows from definitions since there are only finitely many control states). 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 $\mathtt{c}$ 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 $\mathtt{c}$. To check that a is unbounded in channel $\mathtt{c}$, 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 (Lemma 3.14). Hence, in this case, a is unbounded in channel $\mathtt{c}$. □

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 [24] to prove NP hardness for reachability, repeated control state reachability, unboundedness and non-termination.

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

Proof. We reduce from 3SAT. Given a 3-CNF formula $\text{clause}_1 \wedge \cdots \wedge \text{clause}_m$ over variables $x_1, \ldots, x_n$, we construct a flat FIFO machine with n channels $\{x_1, \ldots, x_n\}$. There are two letters $0, 1$ in the message alphabet. The channel x_i is used to keep a guess of the truth assignment to the variable x_i. The flat FIFO machine consists of the gadgets shown in Fig. 3. The gadget for variable x_i adds either 0 (in the top transition) or 1 (in the bottom edge) to channel x_i. At the end of this gadget, channel x_i will have either 0 or 1. We will sequentially compose the gadgets for all variables. Starting from the initial control state of



(a) Gadget for variable x_i (b) Gadget for clause $c_1 = x_1 \vee \neg x_2 \vee x_3$

(c) Gadget for cleaning up variable x_i

FIGURE 3. Gadgets used in the proof of Theorem 3.16

the gadget for variable x_1, we reach the final control state of the gadget for variable x_n and the contents of the channels $x_1, \ldots, x_n$ determine a truth valuation.

The gadget for the example clause $c_1 = x_1 \vee \neg x_2 \vee x_3$ (gadgets for other clauses follow similar pattern) is shown in Fig. 3. The gadget checks that channel x_1 has 1 (in the top path) or that channel x_2 has 0 (in the middle path) or that channel 3 has 1 (in the bottom path). We append the clause gadgets to the end of the variable gadgets one after the other. All clauses are satisfied by the truth valuation determined by the contents of channels $x_1, \ldots, x_n$ iff we can reach the last control state of the last clause.

The gadget for cleaning up variable x_i is shown on the bottom in Fig. 3. We append the cleanup gadgets to the end of the clause gadgets one after the other.

The given 3-CNF formula is satisfiable iff the last control state of the cleanup gadget for variable x_n can be reached with all channels being empty. Hence, this constitutes a reduction to the reachability problem. Note that in the flat FIFO machine constructed above, there are no loops, so all channels are bounded and none of the control states can be visited infinitely often. We add a self loop to the last control state of the cleanup gadget for variable x_n that adds letter 1 to channel x_1. If this loop can be reached, then it can be iterated infinitely often to add unboundedly many occurrences of the letter 1 to channel x_1. Now, the given 3-CNF formula is satisfiable iff the constructed flat FIFO machine is unbounded iff channel x_1 is unbounded iff letter 1 is unbounded in channel x_1 iff there is a non-terminating run iff the last control state of the cleanup gadget for variable x_n can be reached infinitely often. Hence reachability, unboundedness, channel unboundedness, letter channel unboundedness, non-termination and repeated control state reachability are all NP-hard. □

Hence we deduce the main result of this Section.

Theorem 3.17 (Most properties are NP-complete)**.** *For flat FIFO machines, 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. Complexity of Reachability for Flat Lossy FIFO Machines

Let us informally recall that lossy FIFO machines (often called lossy channel systems [1]) are like FIFO machines except that the FIFO semantics allows the lost of any message in the FIFO channels in any configuration. The reachability set on each channel is then downward closed for the subword ordering, hence by Higman's Theorem, one deduces that it is regular; the knowledge of the regularity of the reachability set provides a semi-algorithm for deciding non-reachability by enumerating all inductive forward reachability invariants. By combining this semi-algorithm with a fair exploration of the reachability tree that enumerates all reachable configurations, one obtains an algorithm that decides reachability. Since reachability is Hyper-Ackermann-complete for lossy FIFO machines [35], it is natural to study the complexity of reachability in *flat* lossy FIFO machines.

Abdulla, Collomb-Annichini, Bouajjani and Jonsson studied the verification of lossy FIFO machines by accelerating loops and representing them by a class of regular expressions called Simple Regular Expressions (SRE) [1, 2]. Recall that SRE are exactly regular languages that are downward closed (for the subword ordering). Suppose a lossy FIFO machine (with one channel to simplify notations) has a loop labeled by σ and L is a SRE. Let $\sigma^*(L)$ denote the set of channel contents reachable after executing the loop arbitrarily many times, starting from channel contents that are in L. By analyzing the polynomial (quadratic) algorithms for computing $\sigma^*(L)$, we obtain an upper bound for the computation of the reachability set of a flat lossy FIFO machine.

Theorem 4.1. *The reachability set of a flat lossy FIFO machine S is a SRE that can be computed in exponential time.*

Proof. The SRE for $\sigma^*(L)$ where L is a SRE can be computed in quadratic time [1, Corollary 3]. By iterating this computation on the flat structure, we obtain a SRE of size exponential describing the reachability set. □

From Theorem 4.1, we may deduce that reachability is in EXPTIME for flat lossy FIFO machines. Since there is a linear algorithm for checking whether a SRE is included in another one [1, Lemma 3], we may use this algorithm for checking whether a word $w = w_1 w_2 \cdots w_n$ (where all w_i are letters) is in a SRE L by testing whether the associated SRE $L_w = (w_1 + \epsilon).(w_2 + \epsilon) \ldots (w_n + \epsilon)$ is included in L. This proves that reachability of a configuration (q, w) is in EXPTIME. Let us show now that reachability is in Np for flat lossy FIFO machines.

We first prove that the control-state reachability problem is in Np for front-lossy FIFO machines and then we will use an easy reduction of reachability in flat lossy FIFO machines to the control-state reachability problem for front-lossy FIFO machines.

A FIFO machine is said to be *front-lossy* if at any time, any letter at the front of any channel can be lost. A front-lossy FIFO machine S is a tuple (Q, F, M, Δ) as for standard FIFO machines defined in Definition 2.1. Only the semantics change for front-lossy machines. Suppose $w \in M^*$ is a sequence and c be a channel; let $(w)_{\mathsf{c}}$ denote the channel valuation that assigns w to c and ϵ to all other channels. The front-lossy semantics is given by a transition system T_S as for standard semantics. For every transition $q \xrightarrow{\mathsf{c}?a} q'$ of a front-lossy FIFO machine, the channel valuation $(wa)_{\mathsf{c}} \cdot \mathbf{w}$ results in the transition $(q, (wa)_{\mathsf{c}} \cdot \mathbf{w}) \xrightarrow{\mathsf{c}?a} (q', \mathbf{w})$ 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, as for standard FIFO machines.

We will prove that control state reachability in flat front-lossy FIFO machines is in NP, by adapting a construction from [24]. We reproduce some definitions from [24] to be able to describe our adaptation.

Definition 4.2 [24]**.** A d-head pushdown automaton is a 9-tuple

$$A = \langle S, \Sigma, \$, \Gamma, \Delta_A, \nu, s_0, \gamma_0, S_f \rangle$$

where

(1) S is a finite non-empty set of states,
(2) Σ is the tape alphabet,
(3) \$ is a symbol not in Σ (the endmarker for the tape),
(4) Γ is the stack alphabet,
(5) Δ_A, the set of transitions, is a mapping from $S \times (\Sigma \cup \{\$\} \cup \{\epsilon\}) \times \Gamma$ into finite subsets of $S \times \Gamma^*$,
(6) $\nu : S \to \{1, \ldots, d\}$ is the head selector function,
(7) $s_0 \in S$ is the start state,
(8) $\gamma_0 \in \Gamma$ is the initial pushdown symbol,
(9) $S_f \subseteq S$ is the set of final states.

Intuitively, a d-HPDA has a a finite-state control (S), d reading heads and a stack. All the reading heads read from the same input tape. Each state $s \in S$ in the finite state control reads from the head given by $\nu(s)$ and pops the top of the stack. The transition relation then non-deterministically determines the new control state and the sequence of symbols pushed on to the stack. The read head moves one step to the right on the input tape. The size of a d-HPDA is the number of bits needed to encode it, where the value $\nu(s)$ is specified using binary encoding for every state s. We write MHPDA for the family of d-HPDA for $d \geq 1$.

We write $(s, \gamma) \overset{[\sigma\rangle_i}{\hookrightarrow} (s', w)$ whenever $(s', w) \in \Delta_A(s, \sigma, \gamma)$, where $\nu(s) = i$. Let us fix a d-HPDA $A = \langle S, \Sigma, \$, \Gamma, \Delta_A, \nu, s_0, \gamma_0, F \rangle$. Define $\mathcal{P} = \{p : \{1, \ldots, d\} \to \mathbb{N}\}$. An instantaneous description (ID) of a is a triple $(s, \tau, p, w) \in S \times \Sigma^*\$ \times \mathcal{P} \times \Gamma^*$. An ID (s, τ, p, w) denotes that A is in state s, the tape content is τ, the reading head i is at the position $p(i)$ and the pushdown store content is w. Let $\vdash$ be the binary relation between IDs defined as follows: we have $(s, \tau, p, w\gamma) \vdash (s', \tau, p', ww')$ iff each of the following conditions is satisfied:

(1) $(s, \gamma) \overset{[\sigma\rangle_i}{\hookrightarrow} (s', w')$ where $\nu(s) = i$ and the letter in position $p(i)$ of τ is σ.
(2) $p'(j) = p(j) + 1$ if $j = \nu(s)$ and $p'(j) = p(j)$ otherwise.

Let $\vdash^*$ denote the reflexive transitive closure of $\vdash$.

We say that reading head i is off the tape in the ID (s, τ, p, w) if $p(i)$ is the last position of τ, where the symbol is \$. We say that (s, τ, p, w) is accepting iff $s \in S_F$ and for every $i \in \{1, \ldots, d\}$, the reading head i is off the tape. A tape content $x \in \Sigma^*$ is accepted by A if $(s_0, x\$, p_0, \gamma_0) \vdash^* (s, x\$, p, w)$ where $p_0(i) = 1$ for all $i \in \{1, \ldots, d\}$ and $(s, x\$, p, w)$ is some accepting ID. Let $L(A)$ be the set of words in Σ^* accepted by A.

A bounded expression is a regular expression $\overline{w} = w_1^* \ldots w_n^*$, where each w_i is a non-empty word over Σ. With slight abuse of notation, we also use $\overline{w}$ for the language defined by $\overline{w}$.

Theorem 4.3 [24]**.** *Let $\{A_i\}_{i \in \{1,\ldots,q\}}$ be a family of MHPDA such that A_i is a d_i-HPDA for every $i \in \{1, \ldots, q\}$. Let d be a constant such that $d \geq \max(d_i \mid i \in \{1, \ldots, q\})$. Let*

$\overline{w} = w_1^* \dots w_n^*$ *be a bounded expression. Checking whether* $\cap_{i=1}^{q} L(A_i) \cap \overline{w}$ *is non-empty is in* Np.

We can now state our Np upper bound result for flat front-lossy machines.

Lemma 4.4. *The control state reachability problem for flat front-lossy machines is in* Np.

Proof. Given a flat front-lossy machine (Q, F, M, Δ) with p channels, an initial configuration $(q_0, \mathbf{w}_0)$ and a target control state q, we construct $(p+1)$ 2-HPDAs $A_0, A_{\mathtt{1}}, \dots, A_{\mathtt{p}}$ and a bounded exression $\overline{w}$ over Δ such that some configuration $(q, \mathbf{w})$ is reachable from $(q_0, \mathbf{w}_0)$ iff $\cap_{i=1}^{q} L(A_i) \cap \overline{w}$ is non-empty. We assume without loss of generality that $\mathbf{w}_0(\mathtt{c}) = \epsilon$ for all channels $\mathtt{c}$; if not, we prepend paths that add the required symbols to each channel. Since the given front-lossy machine is flat, there is a bounded expression $\overline{w}$ over Δ whose language is the set of paths from q_0 to q; this is the bounded expression required. Next we describe the 2-HPDAs.

The automaton A_0 simply checks whether the tape content is in the language of $\overline{w}$. This can be done with a single reading head and without a stack. For every channel $\mathtt{c}$, the 2-HPDA $A_{\mathtt{c}}$ will check that the sequence of transitions in its tape is viable, with respect to the contents of channel $\mathtt{c}$. Suppose $M = \{a_1, a_2, \dots, a_n\}$. Figure 4 illustrates $A_{\mathtt{c}}$, which uses the pushdown symbol ζ and two reading heads H and h. A transition labeled $[\{!\} \times M\rangle_H, \epsilon/\zeta$ is to be read as follows: the reading head H can read any transition t in Δ, provided t sends some letter to channel $\mathtt{c}$, nothing is popped from the stack and ζ is pushed on to the stack. At any state of $A_{\mathtt{c}}$, there are self loops that can read any transition in Δ that does not interact with channel $\mathtt{c}$. The self loops are not shown in the figure to reduce clutter.

In state q_H, head H reads all transitions in Δ that is of the form $\mathtt{c}!a_i$ for any $i \in \{1, \dots, n\}$, until a transition of the form $\mathtt{c}?a_i$ for some $i \in \{1, \dots, n\}$ or \$ is read. When a transition of the form $\mathtt{c}?a_i$ is read, control jumps to q_h^i. In q_h^i, the head h looks for the symbol $\mathtt{c}!a_i$, skipping symbols of the form $\mathtt{c}?a_j$ or $\mathtt{c}!a_j$. Intuitively, if a message a_i is to be retrieved from channel $\mathtt{c}$, it should have been sent previously. The head h looks for the transition that sent a_i previously. It skips over any $\mathtt{c}?a_j$ since they don't denote send action. It skips over $\mathtt{c}!a_j$ since in the front-lossy semantics, a message at the front of the queue can be lost. Non-deterministcally, zero or more occurrences of $\mathtt{c}!a_j$ are skipped; then $\mathtt{c}!a_i$ is read and the control is back to q_H. To ensure that the head h does not move beyond H, we use the stack. Whenever H moves, it pushes ζ onto the stack and whenever h moves, it pops ζ from the stack. In any reachable configuration not in the state q_f, $A_{\mathtt{c}}$ maintains the invariant that the number of symbols between h and H is equal to the number of ζ's on the stack. Because of this invariant, head H will be the first to read \$ in which case the control is updated to q_f. Then all the remaining symbols are read by head h until it also reads \$. This way, every retrieve action is matched with a unique send action, so the sequence of actions in the tape of $A_{\mathtt{c}}$ is viable with respect to channel $\mathtt{c}$. Since this is done for every channel, we infer that the control state q is reachable from $(q_0, \mathbf{w}_0)$ iff $\cap_{i=1}^{q} L(A_i) \cap \overline{w}$ is non-empty. Since checking this later conditions is in Np (by Theorem 4.3), we conclude that the control state reachability problem for flat front-lossy systems is in Np. □

There is only a small difference between the construction given in the proof of Lemma 4.4 and the one given in Section IX of [24]. In our constrction, the self loops on $q_h^1, \dots, q_h^n$ in Figure 4 can read $\{?\} \times M$ as well as $\{!\} \times M$, whereas as in [24], only $\{?\} \times M$ can be read.

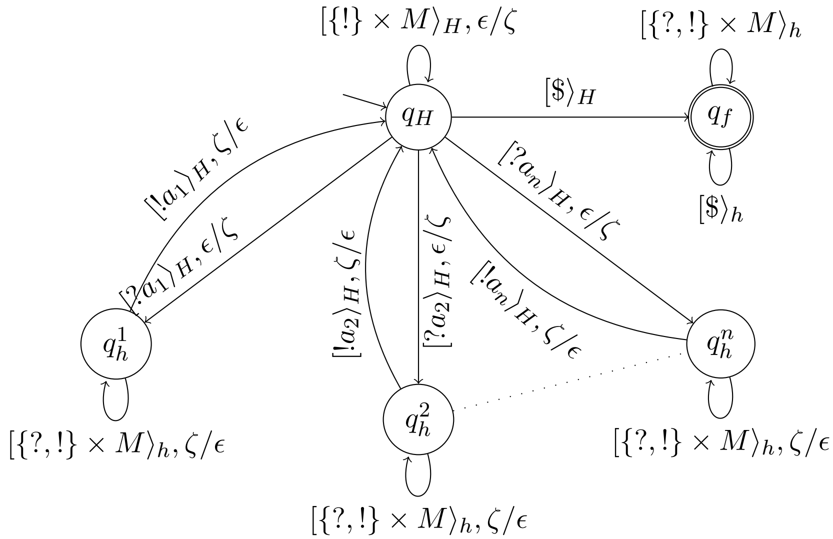


FIGURE 4. The 2-HPDA A_c

Hence we deduce that reachability is in NP for both flat front-lossy FIFO machines and flat lossy FIFO machines. To achieve these results, we reduce reachability (in both models) to control-state reachability in a front-lossy FIFO machine.

To test whether $(q, \mathbf{w})$ is reachable from $(q_0, \mathbf{w}_0)$ in a flat front-lossy FIFO machine S, we complete S into the front-lossy FIFO machine $S_{(q,\mathbf{w})}$ by adding a new path in S (in a similar way as the added path in the proof of Proposition 3.1), from q to q_{stop}, that essentially consumes $\mathbf{w}$. We obtain that $(q, \mathbf{w})$ is reachable in S iff q_{stop} is reachable in the front-lossy FIFO machine $S_{(q,\mathbf{w})}$.

To test whether $(q, \mathbf{w})$ is reachable from $(q_0, \mathbf{w}_0)$ in a flat lossy FIFO machine S', we complete S' into the lossy FIFO machine $S'_{(q,\mathbf{w})}$ by adding a new path in S' similarly as above from q to q_{stop}, such that the added path essentially consumes $\mathbf{w}$. We obtain that $(q, \mathbf{w})$ is reachable in S' iff q_{stop} is reachable in the flat lossy FIFO machine $S'_{(q,\mathbf{w})}$. Then we observe that control state reachability in flat lossy machines reduces to control state reachability in flat front-lossy machines. Hence, reachability in flat lossy FIFO machines reduces to control state reachability in flat front-lossy FIFO machines, which is in NP.

Finally, we can use the same reduction as the one used in the proof of Theorem 3.16 to prove NP-hardness in flat (front-)lossy FIFO machines. The only way to reach the target state in the FIFO machine of that reduction is to not have any losses in the entire operation of the machine, in addition to satisfying the given 3-CNF formula. Hence, the introduction of (front-)lossy semantics will not change anything in the proof of Theorem 3.16. So we may deduce the following.

Theorem 4.5. *Reachability is* NP*-complete for both flat front-lossy FIFO machines and flat lossy FIFO machines.*

During the review process, we were aware of a new unpublished paper from Schnoebelen [36] about flat lossy machines. This paper analyses iterations of lossy channel actions. As a consequence, it also proves the NP upper bound (for reachability and similar problems) by using compressed word techniques.

5. Construction of an Equivalent Counter System

Suppose we want to model check flat FIFO machines against logics in which atomic formulas are of the form $\#^a_{\mathtt{c}} \geq k$, which means there are at least k occurrences of the letter a in channel $\mathtt{c}$. Suppose the letter a denotes an undesirable situation and we would like to ensure that if there are 4 occurrences of the letter a, then the number reduces within the next two steps. This is expressed by the LTL formula $\mathtt{G}(\#^a_{\mathtt{c}} \geq 4 \Rightarrow \mathtt{XX}\neg(\#^a_{\mathtt{c}} \geq 4))$, where $\mathtt{G}$ and $\mathtt{X}$ are the usual LTL operators.

There is no easy way of designing an algorithm for this model checking problem based on the construction in [24], even though we solved reachability and related problems in previous sections using that construction. That construction is based on simulating FIFO machines using automata that have multiple reading heads on an input tape. The channel contents of the FIFO machine 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 [10] 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 machines. This has the additional advantage of being amenable to analysis using existing tools on counter machines. Counter machines 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[2] of constraints of the form $C = 0$ and $C > 0$, where $C \in K$.

Definition 5.1 (Counter machines)**.** A counter machine S is a tuple (Q, K, Δ) 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 (Q, K, Δ) to denote labeled counter machines. The semantics of a counter machine is a transition system with set of states $Q \times \mathbb{N}^K$, called *configurations of the counter machines*. 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 machine, 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 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 machine 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 machine is the disjoint union of $M_{\mathtt{1}}, \ldots, M_{\mathtt{p}}$, where $M_{\mathtt{c}}$ is the alphabet for channel $\mathtt{c}$. In the following, let $S = (Q, F, M, \Delta)$ be a flat FIFO machine, where the set of channels $F = \{1, \ldots, \mathtt{p}\}$ and the set of transitions $\Delta = \{t_1, \ldots, t_r\}$.

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

The counting abstraction machine. The idea behind the *counting abstraction machine* 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 machine 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 machine will decrement the counter (a, t'). The counting abstraction machine doesn't exactly simulate the flat FIFO machine. For example, if the transition labeled $(a, t_1)^{--}$ in Fig. 5(b) 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 machine. We use other counter machines to maintain the order of letters.

Formally, the *counting abstraction machine* corresponding to S is a labeled counter machine $S_{\text{count}} = (Q, K, \Delta_{\text{count}}, \psi, T)$, where $(Q, K, \Delta_{\text{count}})$ is a counter machine 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}:

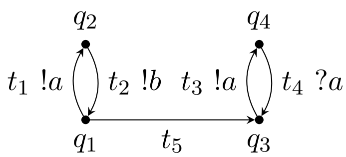
- If t *sends* a message, $t = q_1 \xrightarrow{\mathsf{c}!a} q_2$, then the transition $t_{\text{count}} = q_1 \xrightarrow{(a,t)^{++}} q_2$ is added to Δ_{count}; we define $\psi(t_{\text{count}}) = \tau$ and $T(t_{\text{count}}) = t$.
- If $t = q_1 \longrightarrow q_2$ doesn't change any channel content, 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, $t = q_1 \xrightarrow{\mathsf{c}?a} q_2$, then the set of transitions A_t is added to Δ_{count} with $A_t = \{\delta_{a,t'} = q_1 \xrightarrow[(a,t')>0]{(a,t')^{--}} q_2 \mid t' \text{ sends } a \text{ to channel } \mathsf{c}\}$. 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 machines later and T will be used to match the traces of this counter machine with those of the original flat FIFO machine. 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.

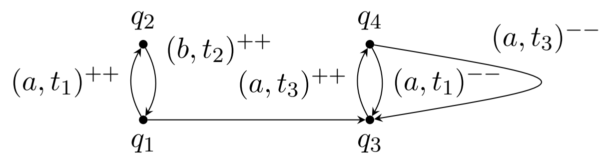
Example 5.2. Figure 5(a) shows a flat FIFO machine and Fig. 5(b) shows its counting abstraction machine.

Note that the counting abstraction machine associated with the flat FIFO machine is not flat. Indeed, the receiving transition $t_4 = q_4 \xrightarrow{?a} q_3$ in the FIFO machine is "translated" into two decrementation transitions $q_4 \xrightarrow{(a,t_1)^{--}} q_3$ and $q_4 \xrightarrow{(a,t_3)^{--}} q_3$ in the counting abstraction machine; these transitions breake the flatness property by creating nested loops on $\{q_3, q_4\}$.

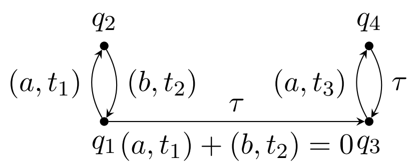
The order machine. The *order machine for channel* c is a labeled counter machine $S^{\mathsf{c}}_{\text{order}} = (Q, K, \Delta^{\mathsf{c}}_{\text{order}}, \psi^{\mathsf{c}})$, where $(Q, K, \Delta^{\mathsf{c}}_{\text{order}})$ is a counter machine and ψ^{c} is a labeling function. The set of control states Q and the set of counters K are the same as in the counting abstraction machine. The set $\Delta^{\mathsf{c}}_{\text{order}}$ of *transitions* of $S^{\mathsf{c}}_{\text{order}}$ and the *labeling function*



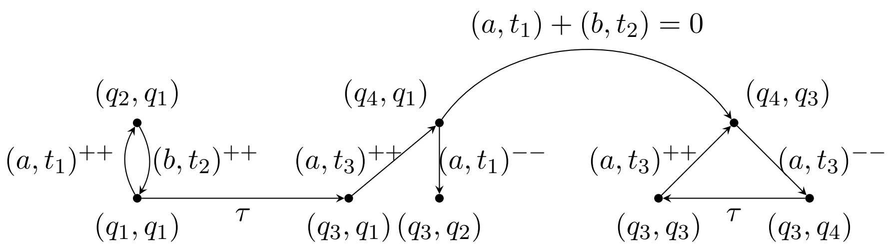
(a) Flat FIFO machine



(b) Counting abstraction machine



(c) Order machine



(d) Synchronized counter system, consisting of the counting abstraction machine (b) and the order machine (c)

FIGURE 5. An example flat FIFO machine (a) and the equivalent counter system (d).

$\psi^{\mathsf{c}} : \Delta^{\mathsf{c}}_{\text{order}} \to (M \times \Delta) \cup \{\tau\}$ are defined as follows: for every $t \in \Delta$, one adds the following transitions in $\Delta^{\mathsf{c}}_{\text{order}}$:

- 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'.

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

This constraint ensures that all the letters produced by iterations of σ are retrieved before letters produced by later transitions.

Figure 5(c) shows the order machine corresponding to the flat FIFO machine of Fig. 5(a).

The synchronized counter system. We will synchronize the counting abstraction machine S_{count} with the order machines $(S^{\mathsf{c}}_{\text{order}})_c$ by *rendez-vous* on transition labels.

Suppose that the machine $S^{\mathsf{c}}_{\text{order}}$ is in state q_2 as shown in Fig. 5(c) and the machine S_{count} is in state q_4, as shown in Fig. 5(b). The machine $S^{\mathsf{c}}_{\text{order}}$ is in state q_2 and the only transition going out from q_2 is labeled by (b, t_2), denoting the fact that the next letter to be retrieved from the channel is b. The machine S_{count} 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 machine $S^{\mathsf{c}}_{\text{order}}$, whose next transition is labeled with (b, t_2). The guard $(a, t_1)+(b, t_2) = 0$ in the bottom transition in Fig. 5(c) 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 machine) or the function ψ^{c} (if the transition is in the order machine for channel c).

The *synchronized counter system* $S_{\text{sync}} = S_{\text{count}} \mid\mid S^{1}_{\text{order}} \mid\mid \cdots \mid\mid S^{\mathsf{c}}_{\text{order}} \mid\mid \cdots \mid\mid S^{\mathsf{p}}_{\text{order}}$ is the synchronized (by rendez-vous) product of the *counting abstraction machine* S_{count} and the *order machines* $S^{\mathsf{c}}_{\text{order}}$ for all channels $\mathsf{c} \in \{1, \dots, \mathsf{p}\}$. All counter machines 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 $Q^{\mathsf{p}+1}$. Transitions labeled with τ need not synchronize with others. Each transition labeled (by the function ψ or ψ^{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 machine and one of the order machines. For a global control state $\overline{q} \in Q^{\mathsf{p}+1}$, $\overline{q}(0)$ denotes the local state of the counting abstraction machine and $\overline{q}(\mathsf{c})$ denotes the local state of the order machine for channel c. The synchronized counter system maintains the channel contents of the flat FIFO machine as explained next.

A weak bisimulation between the FIFO machine and the synchronized system. 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 machine S. The corresponding configuration of S is $h(\overline{q}, \nu) = (\overline{q}(0), h_1(v_1), h_2(v_2), \dots, h_{\mathsf{p}}(v_{\mathsf{p}}))$, where the words $v_{\mathsf{c}} \in \Delta^*$ and morphisms $h_{\mathsf{c}} : \Delta^* \to M^*$ are as follows. Fix a channel c. Let $v_{\mathsf{c}} \in \Delta^*$ be a word labelling a path in S from $\overline{q}(\mathsf{c})$ to $\overline{q}(0)$ such that $Parikh(v_{\mathsf{c}})(t) = \nu\left((a, t)\right)$ for every transition $t \in \Delta$ that sends some letter a to channel c. Now, define $h_{\mathsf{c}}(t) = a$ if t sends some letter a to channel c and $h_{\mathsf{c}}(t) = \epsilon$ otherwise. The word $h_c(v_{\mathsf{c}})$ 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 (recall that a bounded language is included in a language of the form $w_1^* w_2^* \cdots w_k^*$). Intuitively, the path v_{c} gives the order of letters in channel c and the counters give the number of occurrences of each letter. Let us denote by $R_{h,\text{sync}}$ the relation $\{(h((\overline{q}, \nu)), (\overline{q}, \nu)) \mid (\overline{q}, \nu) \text{ is reachable in } S_{\text{sync}}\}$.

Example 5.3. Figure 5(d) shows the reachable states of the synchronized counter system for the flat FIFO machine in Fig. 5(a). Initially, both the counting abstraction machine and the order machine are in state q_1, so the global state is (q_1, q_1). Then the counting abstraction machine may execute the transition labeled $(a, t_1)^{++}$ and go to state q_2 while the order machine stays in state q_1, resulting in the global state (q_2, q_1). Consider the global state (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 $\mathtt{c} = \mathtt{1}$, $v_\mathtt{c} = t_2(t_1t_2)^2t_5t_3t_4$ and $h_\mathtt{c}(v_\mathtt{c}) = b(ab)^2a$.

Let us recall that a relation R between the reachable configurations of the FIFO machine 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 $((q',\mathbf{w}'),(\overline{q'},\nu')) \in R$.

Proposition 5.4. *The relation $R_{h,\text{sync}}$ is a weak bisimulation.*

Proof. Suppose $(h((\overline{q},\nu)),(\overline{q},\nu)) \in R_{h,\text{sync}}$, where $\overline{q}(0) = q$. Suppose there is a transition $h((\overline{q},\nu)) \xrightarrow{t} (q',\mathbf{w}')$ in S. We have $h((\overline{q},\nu)) = (\overline{q}(0),\mathbf{w})$, where $\mathbf{w}(\mathtt{c}) = h_\mathtt{c}(v_\mathtt{c})$ for every channel c and $v_\mathtt{c} \in \Delta^*$ is a word labelling a path in S from $\overline{q}(\mathtt{c})$ to $\overline{q}(0)$ such that $Parikh(v_\mathtt{c})(t) = \nu((a,t))$ for every transition $t \in \Delta$ that sends some letter to channel $\mathtt{c}$ (and a is the letter that is sent by t).

We will prove condition (1) above for weak bisimulation by a case analysis, depending on the type of transition t.

Case 1: transition t is of the form $(q,\mathbf{w}) \to (q',\mathbf{w})$. In S_{sync}, the counting abstraction machine executes the transition $q \to q'$ and the order machines do not perform any transition. Then S_sync is in the configuration $(q',\overline{q}(\mathtt{1}),\dots,\overline{q}(\mathtt{p}),\nu)$, where $\mathtt{p}$ is the number of channels. For every channel $\mathtt{c}$, we get a word v'_c labelling a path in S from $\overline{q}(\mathtt{c})$ to q' such that $Parikh(v_\mathtt{c})(t') = \nu((a,t'))$ for every transition $t' \in \Delta$ that sends some letter to channel $\mathtt{c}$ (and a is the letter that is sent by t') as follows: we simply append the transition $q \to q'$ to v_c. Hence, $h_\mathtt{c}(v'_c) = h_\mathtt{c}(v_c)$ and $((q',\mathbf{w}),(q',\overline{q}(\mathtt{1}),\dots,\overline{q}(\mathtt{p}),\nu)) \in R_{h,\text{sync}}$.

Case 2: transition t is of the form $(q,\mathbf{w}) \xrightarrow{\mathtt{c}!a} (q',\mathbf{w}\cdot\mathbf{a}_\mathtt{c})$ (recall that $\mathbf{a}_\mathtt{c}$ is the channel valuation that assigns a to channel $\mathtt{c}$ and ϵ to all others). In S_sync, the counting abstraction machine executes the transition $q \xrightarrow{(a,t)^{++}} q'$ and the order machines do not execute any transitions. Then S_sync is in the configuration $(q',\overline{q}(\mathtt{1}),\dots,\overline{q}(\mathtt{p}),\nu')$, where ν' is obtained from ν by adding one to the counter (a,t). For every channel $\mathtt{c}'$, we get a word $v'_{c'}$ labelling a path in S from $\overline{q}(\mathtt{c}')$ to q' such that $Parikh(v'_{\mathtt{c}'})(t') = \nu'((a',t'))$ for every transition $t' \in \Delta$ that sends some letter to channel $\mathtt{c}'$ (and a' is the letter that is sent by t') as follows: we simply append the transition $q \xrightarrow{\mathtt{c}!a} q'$ to $v_{\mathtt{c}'}$. Hence, $h_{\mathtt{c}'}(v'_{c'}) = h_{\mathtt{c}'}(v_{c'})$ for $\mathtt{c}' \neq \mathtt{c}$ and $h_\mathtt{c}(v'_c) = h_\mathtt{c}(v_c)\cdot a$. Hence, $((q',\mathbf{w}\cdot\mathbf{a}_\mathtt{c}),(q',\overline{q}(\mathtt{1}),\dots,\overline{q}(\mathtt{p}),\nu')) \in R_{h,\text{sync}}$.

Case 3: transition t is of the form $(q,\mathbf{a}_\mathtt{c}\cdot\mathbf{w}) \xrightarrow{\mathtt{c}?a} (q',\mathbf{w})$. Since $(q,\mathbf{a}_\mathtt{c}\cdot\mathbf{w}) = h(\overline{q},\nu)$, $(\mathbf{a}_\mathtt{c}\cdot\mathbf{w})(c) = h_\mathtt{c}(v_\mathtt{c})$, where $v_\mathtt{c}$ is a word labeling a path in S from $\overline{q}(\mathtt{c})$ to q such that $Parikh(v_\mathtt{c})(t') = \nu((a',t'))$ for every transition $t' \in \Delta$ that sends some letter to channel $\mathtt{c}$ (and a' is the letter that is sent by t'). Hence, the first transition in $v_\mathtt{c}$ that sends a letter to channel c is of the form $t' = q_1 \xrightarrow{\mathtt{c}!a} q_2$ and $\nu((a,t')) \geq 1$. In S_sync, the order machine for channel $\mathtt{c}$ executes the sequence of transitions from $\overline{q}(\mathtt{c})$ to q_2; note that $\psi_\mathtt{c}$ labels the last transition of this sequence with (a,t') and labels other transitions in this sequence with τ. The order machines for other channels do not execute any transitions. The counting abstraction machine executes the transition $q \xrightarrow[(a,t')>0]{(a,t')^{--}} q'$, which is labeled by ψ with (a,t')

so it can synchronize with the transition $q_1 \longrightarrow q_2$ executed by the order machine for channel c. Now the synchronized counter system S_{sync} is in the configuration $(\overline{q'}, \nu')$, where $\overline{q'}$ is obtained from $\overline{q}$ by changing $\overline{q}(0)$ from q to q' and changing $\overline{q}(\mathtt{c})$ to q_2 and ν' is obtained from ν by subtracting one from the counter (a, t'). For channels $\mathtt{c}' \neq \mathtt{c}$, let $v'_{\mathtt{c}'} = v_{\mathtt{c}'}$ and let $v'_{\mathtt{c}}$ be obtained from $v_{\mathtt{c}}$ by removing the prefix ending at q_2. Now for every channel $\mathtt{c}'$, the word $v'_{c'}$ labels a path in S from $\overline{q'}(\mathtt{c}')$ to q' 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'). Hence, $((q', \mathbf{w}), (\overline{q'}, \nu')) \in R_{h,\text{sync}}$ (end of Case 3 and of condition (1)).

Next we prove condition (2) for weak bisimulation: for every transition $(\overline{q}, \nu) \xrightarrow{t_s} (\overline{q'}, \nu')$ in S_{sync} with $T(t_s) = \tau$, we will show that $((q, \mathbf{w}), (\overline{q'}, \nu')) \in R_{h,\text{sync}}$. Recall that the labeling function T of S_{count} is extended 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 $T(t_s) = \tau$. Hence, if S_{sync} executes a transition $(\overline{q}, \nu) \xrightarrow{t_s} (\overline{q'}, \nu')$ and $T(t_s) = \tau$, the counting abstraction machine does not participate in t_s. The only transition participating in t_s is some transition $q_1 \to q_2$ in $S^{\mathtt{c}}_{\text{order}}$ for some channel $\mathtt{c}$ satisfying the following property: $q_1 \xrightarrow{x} q_2$ is a transition in the FIFO machine S and x does not contain any sending operation of any letter to $\mathtt{c}$. In this case, S_{sync} goes to the configuration $(\overline{q'}, \nu')$ where $\nu' = \nu$ and $\overline{q'}$ is obtained from $\overline{q}$ by changing $\overline{q}(\mathtt{c})$ from q_1 to q_2. For channels $\mathtt{c}' \neq \mathtt{c}$, let $v'_{\mathtt{c}'} = v_{\mathtt{c}'}$ and let $v'_{\mathtt{c}}$ be obtained from $v_{\mathtt{c}}$ by removing the prefix transition $q_1 \xrightarrow{x} q_2$. Now for every channel $\mathtt{c}'$, the word $v'_{c'}$ labels 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'). Hence, $(h((\overline{q}, \nu)), (\overline{q'}, \nu')) \in R_{h,\text{sync}}$.

Next we prove condition (3) for weak bisimulation by a case analysis, depending on the type of transition t_s.

Case 1: the transition t_s is of the form $t_{\text{count}} = q \longrightarrow q'$ executed by the counting abstraction machine. Then S_{sync} goes to the configuration $(q', \overline{q}(\mathtt{1}), \ldots, \overline{q}(\mathtt{p}), \nu)$. If $h((\overline{q}, \nu)) = (q, \mathbf{w})$, then the FIFO machine S executes the transition $q \longrightarrow q'$ and we conclude that $((q', \mathbf{w}), (q', \overline{q}(\mathtt{1}), \ldots, \overline{q}(\mathtt{p}), \nu)) \in R_{h,\text{sync}}$ as in case 1 above.

Case 2: the transition t_s is of the form $t_{\text{count}} = q \xrightarrow{(a,t)^{++}} q'$ executed by the counting abstraction machine, where $t = q \xrightarrow{\mathtt{c}!a} q'$ is a transition of the FIFO machine S. Then S_{sync} goes to the configuration $(q', \overline{q}(\mathtt{1}), \ldots, \overline{q}(\mathtt{p}), \nu')$, where ν' is obtained from ν by adding one to the counter (a, t). If $h((\overline{q}, \nu)) = (q, \mathbf{w})$, then the FIFO machine S executes the transition $q \xrightarrow{\mathtt{c}!a} q'$ and goes to the configuration $(q, \mathbf{w} \cdot \mathbf{a}_{\mathtt{c}})$. We conclude that $((q, \mathbf{w} \cdot \mathbf{a}_{\mathtt{c}}), (q', \overline{q}(\mathtt{1}), \ldots, \overline{q}(\mathtt{p}), \nu'))$ as in case 2 above.

Case 3: the transition t_s is a synchronized transition with the counting abstraction machine executing the transition $\delta_{a,t'} = q \xrightarrow[(a,t')>0]{(a,t')^{--}} q'$ (where t' sends the letter a to channel $\mathtt{c}$) and the order machine for channel $\mathtt{c}$ executing the transition $q_1 \to q_2$ (which is labeled with (a, t') by $\psi_{\mathtt{c}}$). The synchronized transition system S_{sync} goes to the configuration $(\overline{q'}, \nu')$, where $\overline{q'}$ is obtained from $\overline{q}$ by changing $\overline{q}(0)$ from q to q' and changing $\overline{q}(\mathtt{c})$ to q_2 and ν' is obtained from ν by subtracting one from the counter (a, t'). If $h((\overline{q}, \nu)) = (q, \mathbf{a}_{\mathtt{c}} \cdot \mathbf{w})$, then the FIFO machine S executes the transition $(q, \mathbf{a}_{\mathtt{c}} \cdot \mathbf{w}) \xrightarrow{\mathtt{c}?a} (q', \mathbf{w})$. We conclude that $((q', \mathbf{w}), (\overline{q'}, \nu')) \in R_{h,\text{sync}}$ as in case 3 above. □

A bisimulation between the FIFO machine and the modified synchronized system. We proved weak bisimulation above instead of bisimulation, due to the presence of silent transitions in the order machines participating in S_{sync}. We can modify the order machines as follows to get a bisimulation. For every channel c and every transition $q_1 \longrightarrow q_2$ labeled τ in $S^{\mathsf{c}}_{\text{order}}$, remove the transition and merge the two states q_1, q_2 into one state. If exactly one of the two states q_1, q_2 was an anchor state, retain the name of the anchor state as the name of the merged state. Otherwise, retain q_2 as the name of the merged state. Repeat this process until there are no more transitions labeled τ. Note that we have only removed transitions that do not correspond to any transition of S sending letters to channel c. Such transitions are assigned ϵ by the morphism h_{c} defined in the paragraph preceding Ex. 5.3. Hence, the deletion of τ-labeled transitions do not affect the correspondence between the configurations of S and S_{sync}. If there are no sending transitions between two anchor states, the above deletion procedure may result in two anchor states getting merged, destroying the flatness of the order machine. Next we describe a way to tackle this.

Suppose a transition t' in the order machine modified as above corresponds to a transition t in the original flat FIFO machine S. Suppose this transition t of S is in a loop ℓ, which is labeled by the sequence of transitions σ. For every transition t_1 in S outside ℓ but reachable from states in ℓ, we make the following modification. If the order machine has a transition t'_1 corresponding to t_1, we add the following guard to t'_1.

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

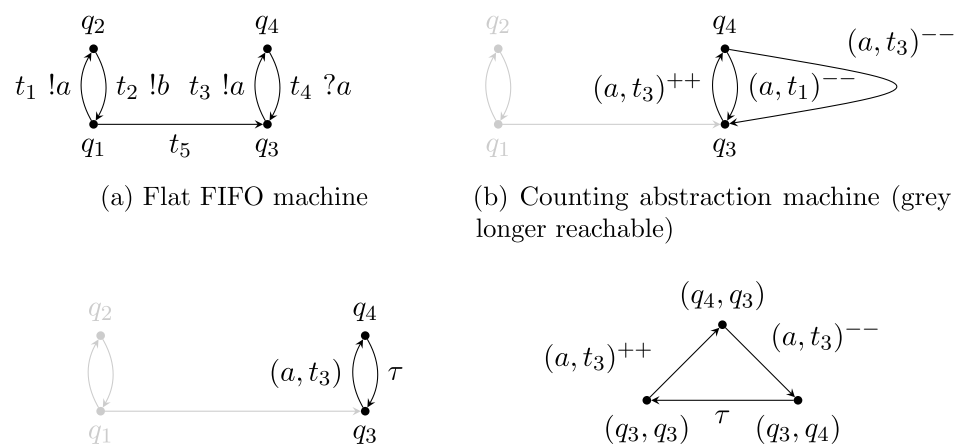
These guards ensure that all letters sent by transitions in ℓ are retrieved before retrieving letters sent by later transitions. In addition, the guards ensure that the modified order machine is flattable. Suppose the loop ℓ in S corresponds to loop ℓ' in $S^{\mathsf{c}}_{\text{order}}$. If a transition occurring after and outside the loop ℓ' is fired in $S^{\mathsf{c}}_{\text{order}}$, loop ℓ' can't be entered again. The reason is that any transition t'' in the loop ℓ' tries to decrement some counter (a, t''), but it can't be decremented since it has value 0, as checked in the guard newly added to every transition occurring after ℓ'.

The modified order machines don't have τ-labeled transitions anymore, hence the modified synchronized counter system S'_{sync} doesn't have silent transitions. Now a proof similar to that of Proposition 5.4 can be used to show bisimulation between S and the modified synchronized counter system S'_{sync}.

Let $R'_{h,\text{sync}}$ be the relation $\{(h((\overline{q}, \nu)), (\overline{q}, \nu)) \mid (\overline{q}, \nu) \text{ is reachable in } S'_{\text{sync}}\}$.

Proposition 5.5. *The relation $R'_{h,\text{sync}}$ is a bisimulation.*

Trace-flattening. The counting abstraction machine S_{count} is not flat in general. E.g., there are two transitions from q_4 to q_3 in Fig. 5(b). 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 machine and states q_3, q_4 of the order machine as shown in Fig. 6 (parts of the system that are no longer reachable are greyed out). Now the transition labeled $(a, t_1)^{--}$ can't be used and the run is as shown in Fig. 6(d), which is a flat counter machine. In general, suppose $\ell_0, \ell_1, \ldots, \ell_r$ are the loops in S. There is a flat counter machine 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 machine are in some



(a) Flat FIFO machine

(b) Counting abstraction machine (grey part no longer reachable)

(c) Order machine (grey part no longer reachable)

(d) Part of synchronized counter system still reachable

Figure 6. Flattening

loop ℓ_i and for every channel $\mathtt{c}$, all local states of the order machine $S^{\mathtt{c}}_{\text{order}}$ are in some loop $\ell_{\mathtt{c}}$. 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 machine 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} [18, 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} [18, Definition 8] if S' is a f-flattening of S_{sync} and traces$(S_{\text{sync}}) = f(\text{traces}(S'))$.

Proposition 5.6. *The synchronized counter system S_{sync} is trace-flattable.*

Proof. Starting from a global state $\overline{q}$ of S_{sync}, we claim that we can build a flat counter machine that is a trace-flattening of S_{sync}. Let n_0 be the number of loops in S reachable from $\overline{q}(0)$. For each channel $\mathtt{c}$, let $n_{\mathtt{c}}$ be the number of loops in S reachable from $\overline{q}(\mathtt{c})$. We prove the claim by induction on the vector $\langle n_0, n_1, \ldots, n_{\mathtt{p}}\rangle$. The order on vectors is component-wise comparison — $\langle n_0, n_1, \ldots, n_{\mathtt{p}}\rangle < \langle n'_0, n'_1, \ldots, n'_{\mathtt{p}}\rangle$ if $n_i \le n'_i$ for all $i \in \{0, \ldots, \mathtt{p}\}$ and $n_j < n'_j$ for some $j \in \{0, \ldots, \mathtt{p}\}$.

For the base case, $\langle n_0, n_1, \ldots, n_{\mathtt{p}}\rangle = \mathbf{0}$. From such a global state, the counting abstraction machine and order machines for all the channels have unique paths to follow and hence there is a unique run of S_{sync}. This unique run can be easily simulated by a flat counter machine, proving the base case.

For the induction step, suppose ℓ_0 is the first loop in S reachable from $\overline{q}(0)$ and for every channel $\mathtt{c}$, suppose $\ell_{\mathtt{c}}$ is the first loop in S reachable from $\overline{q}(\mathtt{c})$, with ℓ'_c being the corresponding loop in $S^{\mathtt{c}}_{\text{order}}$. There is a flat counter machine S_{flat} described in the paragraph preceding this lemma, which can simulate runs of the synchronized counter system as long as the counting abstraction machinew doesn't exit the loop ℓ_0 and for every channel $\mathtt{c}$, the order machine $S^{\mathtt{c}}_{\text{order}}$ doesn't exit the loop $\ell'_{\mathtt{c}}$. If the counting abstraction machine exits the loop ℓ_0 (or the order machine $S^{\mathtt{c}}_{\text{order}}$ exits the loop $\ell'_{\mathtt{c}}$ for some channel $\mathtt{c}$), then the vector $\langle n_0 - 1, n_1, \ldots, n_{\mathtt{p}}\rangle$ (or the vector $\langle n_0, n_1, \ldots, n_{\mathtt{c}} - 1, \ldots, n_{\mathtt{p}}\rangle$) is strictly smaller than the vector $\langle n_0, n_1, \ldots, n_{\mathtt{p}} \rangle$.[3] The induction hypothesis shows that there is a flat counter machine S'_{flat} that can cover the remaining possible runs. We sequentially compose S_{flat} and S'_{flat} by identifying the initial state of S'_{flat} with the state of S_{flat} in which the counting abstraction machine exits the loop ℓ_0 (or the order machine $S^{\mathtt{c}}_{\text{order}}$ exits the loop $\ell'_{\mathtt{c}}$). There are finitely many possibilities of the counting abstraction machine or one of the order machines exiting a loop; for each of these possibilities, the induction hypothesis gives a flat counter machine S'_{flat}. We sequentially compose S_{flat} with all such flat counter machines S'_{flat}. The result is a trace-flattening of the synchronized counter system. □

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. In theory, problems on flat FIFO machines can be solved by using tools on counter machines (bisimulation preserves CTL* and trace-flattening preserves LTL [18, Theorem1]); hence we deduce:

Theorem 5.7. *LTL is decidable for flat FIFO machines.*

The decidability of CTL* is an open problem for bisimulation-flattable counter machines [18], so we cannot use it for deciding CTL* in flat FIFO machines.

6. Conclusion and Perspectives

We answered the complexity of the main reachability problems for flat (perfect, lossy and front-lossy) FIFO machines which are NP-complete as for flat counter machines. We also show how to translate a flat FIFO machine into a trace-flattable counter system. This opens the way to model-check a general FIFO machine by *enumerating its flat sub-machines.*

Let us recall the spirit of many tools for non-flat counter machines like FAST, FLATA, ... [4, 6, 5, 13, 22] and for general well structured transition systems [26]. The framework for underapproximating a non-flat machine M proposes to enumerate a (potentially) infinite sequence of flat sub-machines $M_1, M_2, \ldots, M_n, \ldots$, to compute the reachability set of each flat sub-machine M_n, and to iterate this process till the reachability set is computed. For this strategy, we use a fair enumeration of flat sub-machines, which means that every flat sub-machine will eventually appear in the enumeration.

Suppose M_n is a flat FIFO sub-machine enumerated and we want to check if $\mathit{Reach}(M_n)$ is stable under Post_M. We don't want to compute directly $\mathit{Reach}(M_n)$ but we will compute $\mathit{Reach}(C_n)$ that is possible since C_n is a flat counter machine. If there is a transition t in the non-flat machine M that does not have any copy in M_n then M_n is not stable and we continue. Otherwise, transition t of M has copies $t_1, \ldots, t_m$ in M_n. Check if M_n is stable under each transition $t_1, \ldots, t_m$; this is done by testing whether, for every $i = 1, \ldots, m$, $\mathit{Post}_{T_i}(\mathit{Reach}(C_n)) \subseteq \mathit{Reach}(C_n)$ where T_i is the set of transitions, in C_n, associated (by bisimulation) with transition t_i in M_n. If one of the m tests fails, M_n is not stable. Otherwise, M_n is stable.

The following semi-algorithm gives an overview of a strategy to compute the reachability relation and then verify, for instance, whether a configuration is reachable from another one.

- start fairly enumerating flat sub-machines $M_1, M_2, \ldots, M_n, \ldots$
- for every flat subsystem M_n
  - compute the synchronized counter system C_n associated with S_n

[3]This step fails in non-flat FIFO machines; if a loop is exited in a non-flat FIFO machine, it may be possible to reach the loop again, so the vector doesn't necessarily decrease.

– compute the reachability set $Reach(C_n)$
– test whether $Reach(M_n)$ is stable under $Post_M$
– if $Reach(M_n)$ is stable under $Post_M$ we can terminate. Otherwise, we go to the next flat subsystem M_{n+1} and repeat.

The above semi-algorithm terminates if there is a flat FIFO sub-machine having the same reachability set as the entire machine.

But real systems of FIFO systems are often not reduced to an *unique* FIFO machine. Let us show how results on flat FIFO machines can be used to verify *systems* of communicating FIFO machines.

Let us consider a peer-to-peer FIFO system $S = (M_1, M_2, \ldots, M_k)$ where machine M_i communicates with machine M_j through two one-directional FIFO channels: M_i sends letters to M_j through channel $c_{i,j}$ and M_i receives letters from M_j through channel $c_{j,i}$ for every $i, j = 1, \ldots, k$ $(i \neq j)$. Remark that peer-to-peer flat FIFO systems don't produce (by product) a flat FIFO machine. If we consider the product of the three flat machines shown in Fig. 1, the resulting FIFO machine is not flat. It does become flat if we remove the self loop labeled $\texttt{pq}?y$ in Process P. The resulting flat sub-machine is unbounded, so it implies that the original system is also unbounded. Hence, even if the given flat FIFO system don't produce (by product) a flat FIFO machine, some questions can often be answered by analyzing sub-systems and flat sub-machines. Fortunately, reachability in such peer-to-peer flat FIFO systems reduces to reachability in VASS [9], hence the reachability problem is decidable (but with the non-elementary complexity of reachability in VASS).

When systems of FIFO machines $S = (M_1, M_2, \ldots, M_k)$ are not composed of flat FIFO machines, we may use different strategies: we may compute the product M of machines M_i and enumerate the flat sub-machines of M or enumerate the flat sub-systems S_n of S and analyse them.

It remains to be seen if tools can be optimized to make verifying FIFO machines work in practice. This strategy has worked well for counter machines and offers hope for FIFO machines. We have to evaluate all these possible verification strategies on real case studies.

Acknowledgements. We would like to thank the anonymous referees of both the Conference CONCUR'2019 and the Journal LMCS for their attentive reading and their constructive questions and suggestions that allowed us to improve the quality of our article.

References

[1] Parosh Aziz Abdulla, Ahmed Bouajjani, and Bengt Jonsson. On-the-fly analysis of systems with unbounded, lossy FIFO channels. In *CAV*, volume 1427 of *Lecture Notes in Computer Science*, pages 305–318. Springer, 1998.

[2] 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.

[3] 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.

[4] 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.

[5] Sébastien Bardin, Alain Finkel, Jérôme Leroux, and Laure Petrucci. FAST: Acceleration from theory to practice. *International Journal on Software Tools for Technology Transfer*, 10(5):401–424, October 2008.
[6] 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.
[7] Bernard Boigelot. Domain-specific regular acceleration. *STTT*, 14(2):193–206, 2012.
[8] 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.
[9] Benedikt Bollig, Alain Finkel, and Amrita Suresh. Bounded reachability problems are decidable in FIFO machines. In Igor Konnov and Laura Kovacs, editors, *Proceedings of the 31st International Conference on Concurrency Theory (CONCUR'20)*, volume 171 of *Leibniz International Proceedings in Informatics*, Vienna, Austria, September 2020. Leibniz-Zentrum für Informatik. To appear.
[10] 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.
[11] 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.
[12] Marius Bozga, Radu Iosif, and Filip Konecný. Safety problems are np-complete for flat integer programs with octagonal loops. *CoRR*, abs/1307.5321, 2013.
[13] 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.
[14] Daniel Brand and Pitro Zafiropulo. On communicating finite-state machines. *J. ACM*, 30(2):323–342, 1983.
[15] 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.
[16] Gérard Cécé and Alain Finkel. Verification of programs with half-duplex communication. *Information and Computation*, 202(2):166–190, November 2005.
[17] 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.
[18] 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.
[19] 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.
[20] 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.
[21] Stéphane Demri, Amit Kumar Dhar, and Arnaud Sangnier. Taming past LTL and flat counter systems. *Inf. Comput.*, 242:306–339, 2015.
[22] 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.

[23] Frank Drewes and Jérôme Leroux. Structurally cyclic petri nets. *Logical Methods in Computer Science*, 11(4), 2015.

[24] 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.

[25] 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.

[26] Alain Finkel and Jean Goubault-Larrecq. Forward analysis for WSTS, part II: Complete WSTS. *Logical Methods in Computer Science*, 8(3:28), September 2012.

[27] 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.

[28] 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.

[29] Blaise Genest, Dietrich Kuske, and Anca Muscholl. On communicating automata with bounded channels. *Fundam. Inform.*, 80(1-3):147–167, 2007.

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

[31] 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.

[32] Thierry Jéron and Claude Jard. Testing for unboundedness of FIFO channels. *Theor. Comput. Sci.*, 113(1):93–117, 1993.

[33] Julien Lange and Nobuko Yoshida. Verifying asynchronous interactions via communicating session automata. *CoRR*, abs/1901.09606, 2019.

[34] Christos H. Papadimitriou. On the complexity of integer programming. *J. ACM*, 28(4):765–768, October 1981.

[35] Sylvain Schmitz. Complexity hierarchies beyond elementary. *TOCT*, 8(1):3:1–3:36, 2016.

[36] Philippe Schnoebelen. On flat lossy channel machines. *CoRR*, abs/2007.05269, 2020.

[37] Gregoire Sutre. Personal communication, 2018.

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

This work is licensed under the Creative Commons Attribution License. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/ or send a letter to Creative Commons, 171 Second St, Suite 300, San Francisco, CA 94105, USA, or Eisenacher Strasse 2, 10777 Berlin, Germany