Termination, Boundedness and Reachability for Input-bounded FIFO Machines

Amrita Suresh
Supervised by Prof. Alain Finkel
Laboratoire Spécification et Vérification, ENS Paris-Saclay

August 20, 2019

General Context

Software systems are becoming increasingly more concurrent and distributed. Asynchronous distributed systems are widely used from web services, to multi-core architectures. However, error detection is not easy in such systems, especially because the executions are difficult to reproduce and channel contents could be unbounded. In general, we consider finite machines communicating via unbounded channels (henceforth called FIFO systems). These models have the power of Turing machines, which leads to undecidability results in their verification [3]. One way of analysing such systems would be to find approximations which have decidable results and are hence, easier to verify. We could then test properties on these approximations in order to get a semi-decidable results on the system.

Research Problem

My internship focuses on an under-approximation of FIFO systems, known as "input-bounded FIFO systems", which restricts the send operations on every channel to a bounded language, i.e. a language of the form $w_1^* w_2^* ... w_k^*$. However, this type of system has not been extensively studied in the past, while restrictions of the approximation have been studied in [11, 13, 5]. The aim of my internship was to analyse these systems, and see if we can obtain decidability results on the boundedness, termination, and reachability of such systems.

Proposed Contributions

We were able to provide decidability results for the termination and boundedness of input-bounded FIFO systems. In order to do so, we introduced the notion of *branch well structured transition systems* (branch WSTS). This is a generalisation of the idea of well-structured transition systems [8]. We showed that the conditions for branch WSTS are sufficient conditions for verifying the boundedness and termination of the system.

We also found that increasing the power of input-bounded FIFO systems slightly, and instead considering FIFO systems whose *channel contents* are always a bounded language, leads us back to undecidability. We showed that such machines can simulate Minsky machines.

Furthermore, we proved that reachability is decidable in a subclass of input bounded FIFO systems, where every word w_i is just a letter, i.e. input-letter bounded FIFO systems. We further conjecture that using a similar construction, reachability is decidable for the entire class of input-bounded FIFO systems and thus, input-bounded FIFO systems have decidable verification properties.

Arguments Supporting its Validity

The questions of boundedness, termination, and reachability of input-bounded FIFO systems have not been known before. The results of this internship show that they can be promising under-approximations for the general aynchronous distributed processes communicating using perfect queues. Furthermore, we introduce a new generalisation of well-structured systems which can analyse the boundedness and termination of systems which are not well-structured, but still share some properties with such systems. These underapproximations can be practically used to verify negative results for the class of asynchronous distributed processes. Furthermore, from a theoretical perspective, they show that the class we consider are still very powerful, since a slight modification leads us back to undecidability.

Summary and Future Work

The approach of understanding FIFO systems using over and under-approximations is a interesting way forward. We have some interesting subclasses which have decidable properties, and verification of general FIFO systems may be possible by using these approximations using a semi-decidable algorithm.

Another interesting approach forward would be to analyse the complexity of these problems. If they have efficient solutions, we could implement these solutions to partially analyse the system. It would also be interesting to further explore the generalizations of WSTS that we have defined. Another open problem is whether the problem of coverability is decidable for input-bounded FIFO systems.

Acknowledgements

I would like to, first and foremost, thank my advisor Prof. Alain Finkel for his infinite patience, keen insights, and unending support. My stay at LSV was very fruitful largely because of the constructive and educational discussions I had with him. Furthermore, I would like to thank Dr. Benedikt Bollig and Dr. Philippe Schnoebelen for their time, and for enriching my internship experience. I would also like to thank all the members of the LSV, including the PhD students and the guests, for fostering a wonderful research environment with interesting and informative discussions. I would also like to thank Mrs. Imane Mimouni and Mr. Hugues Moretto-Viry for the administrative and technical support. Finally, I would like to thank Prof. Olivier Carton and Prof. Gilles Dowek for all the support during the course of the program.

1 Introduction

Asynchronous distributed processes communicating using First In First Out (FIFO) channels are being widely used for distributed and concurrent programming, and more recently, for web service choreographies. Since such machines simulate Turing machines, most reachability properties are undecidable for such systems. For example, testing the unboundedness of a channel is undecidable [3].

FIFO subclasses with a decidable reachability problem are few, as the model is very powerful. In fact, FIFO systems with a single channel can simulate Turing machines. However, there have been a few decidability results on some subclasses.

Decidable subclasses Half-duplex systems [4] (with two processes) have been known to have certain decidable properties, but the natural extension to three processes leads to undecidability. Apart from this, lossy FIFO systems [1], which are an overapproximation of the general FIFO behaviour (where the channels are no longer perfect) have been shown to be well-structured.

Apart from these, existentially-bounded deadlock-free FIFO automata [12] and synchronisable FIFO systems [10] have decidable properties, but it is undecidable to check if a system is existentially bounded or has the sychronisability property.

Some other methods such as bounded exploration have been employed to investigate this problem further. A recent technique [14] combines a bounded exploration with a convergence test: if the sequence of certain abstractions of the reachable states, for increasing queue bounds, converges, the property is provable. In [7], the authors prove that the control-state reachability modulo bounded languages is decidable for such systems.

Input-bounded FIFO systems Another interesting subclass of FIFO systems are FIFO systems with a bounds on the input-language of the system. The reachability for different-letter-bounded FIFO systems were shown to be reducible to reachability in a special class of Petri nets (with structured terminal markings) [5]. Furthermore, the termination of input-letter bounded FIFO systems was shown to decidable in [13]. However, apart from these results, little has been explored regarding these systems - testing whether a given system is input-bounded is believed to be undecidable. Hence, a particularly interesting subclass of input-bounded FIFO systems, known as *flat FIFO systems* has been of interest. It was shown that various reachability problems are NP-complete for such systems [11].

Proposed contributions We introduce a generalisation on well-structured transition systems which is sufficient to verify boundedness and termination of input-bounded FIFO systems. We also show that reachability is decidable for a subclass of such systems, namely, input-letter-bounded FIFO systems, and conjecture that it holds for input-bounded systems as well. We also show that slight extensions of the model lead back to undecidability.

2 Preliminaries

A binary relation $\leq$ over a set A is a *quasi ordering* (qo) iff it is reflexive and transitive. A quasi ordering $\leq$ is a *partial ordering* iff it is antisymmetric, i.e $a \leq b$ and $b \leq a \implies a = b$. When $\leq$ is a partial ordering then $a < b$ is equivalent to $a \leq b$ and $a \neq b$.

Definition 2.1. A *well quasi ordering* (wqo) $\leq$ over a set A is a qo such that every infinite sequence $x_0, x_1, x_2, ...$ over A contains an *increasing pair* : $\exists i < j$ s.t. $x_i \leq x_j$.

There is an equivalent characterisation of wqos from a result by Erdős and Rado.

Proposition 2.1 ([6])**.** *Assume $(A, \leq)$ is a wqo. Then every infinite sequence in A has an infinite increasing subsequence.*

For example, the set of natural numbers $\mathbb{N}$, along with the standard ordering $\leq$ over it, is a well quasi ordering. On the other hand, the prefix ordering of words over an alphabet Σ, denoted by $\preceq$, is not a well quasi ordering. Consider the alphabet $\Sigma = \{a, b\}$. There exists an infinite antichain $b, ab, aab, ...$ of the prefix ordering.

A *transition system* is defined as $\mathcal{S} = \langle S, \rightarrow \rangle$, where S is a set of states, and $\rightarrow \subseteq S \times S$. We write $s \rightarrow s'$ for $(s, s') \in \rightarrow$. Let $\xrightarrow{*}$ be the transitive and reflexive closure of the relation $\rightarrow$. We write $Post_{\mathcal{S}}(s) = \{s' \in S | s \rightarrow s'\}$ for the set of immediate successors of the state s. $\mathcal{S}$ is finitely branching if for all $s \in S$, $Post_{\mathcal{S}}(s)$ is finite. The *reachability set* of a transition system $\langle S, \rightarrow \rangle$ from an initial state s_0 is $Post^*_{\mathcal{S}}(s_0) = \{s \in S | s_0 \xrightarrow{*} s\}$.

Definition 2.2 (Ordered transition system)**.** An *ordered transition system* $\mathcal{S} = (X, \rightarrow, \leq)$ consists of a wqo $(X, \leq)$ and a transition relation $\rightarrow \subseteq X \times X$.

We now define the notion of *well-structured transition systems* (WSTS).

Definition 2.3. A *well structured transition system* $\mathcal{S} = (S, \rightarrow, \leq)$ is a transition system endowed with a wqo $\leq \subseteq S \times S$ that satisfies a *strong compatibility* condition:

$$s \rightarrow s' \wedge t \geq s \implies \exists t' \geq s', t \rightarrow t'. \tag{1}$$

3 Branch WSTS

For termination and boundedness of infinite transition systems, we see that we only need to consider configurations which are reachable from one other. Hence, we generalize the definition of wqo and compatibility such that the ordering only considers configurations along a run of the system. We shall see that this definition can aid in deciding termination and boundedness of systems which may, in general, not be well structured or well quasi ordered.

The whole behaviour of a transition system $\mathcal{S}$ can be described as a directed (possibly infinite) labelled tree called the reachability tree. We first define this tree.

Definition 3.1 (Reachability tree)**.** Given a transition system $\mathcal{S}$ with the initial state x_0, the reachability tree is defined by the following rules.

- the root n_0 is labelled with the initial state x_0,
- all the nodes are labelled with a reachable state, we will use n to describe the node corresponding to state s, or $n(x)$
- edges are labelled with fireable actions.

Using this definition, we introduce the notion of branch-wqo and branch-compatibility. A *branch* refers to one path starting from the initial configuration.

Definition 3.2. An ordered transition system $\mathcal{S} = (X, \rightarrow, \leq)$ is *branch-well quasi ordering* (branch-wqo) if every infinite labeled branch $n_0(x_0) \rightarrow n_1(x_1) \rightarrow n_2(x_2), ...$ of the reachability tree of $\mathcal{S}$ is wqo.

From Proposition 2.1, it follows that in a branch-wqo, every infinite labeled branch has an infinite strictly increasing sequence.

A wqo over X is a branch-wqo on every tree labeled by X. The converse is false, consider the tree where each branch of length $n+1$ is labeled by $a^n b$. The system is not wqo under the prefix ordering, but the tree is branch-wqo.

We modify the definition of *(strong) compatibility*, and extend it to *(strong) branch-compatibility*, which only considers the (strong) compatibility of states which are reachable from one another.

Definition 3.3 (Strong branch-compatibility)**.** Let S be an ordered transition system $S = (X, \rightarrow, \leq)$. For all configurations s, t, s' such that $s \leq t$ and for all $w \in \Sigma^*$ and for all $a \in \Sigma$, $s \xrightarrow{a} s' \xrightarrow{w} t$ implies that there exists a configuration t' such that $t \xrightarrow{a} t'$ and $s' \leq t'$.

And using these two modified definitions, we have the following notion of branch WSTS.

Definition 3.4 (Branch WSTS)**.** An ordered transition system $\mathcal{S} = (S, \rightarrow, \leq)$ is *branch WSTS* (resp. with strong monotony) if $\mathcal{S}$ is (resp. strongly) branch-compatible and branch-wqo.

Definition 3.5 (Strict branch-compatibility)**.** Let S be a branch WSTS $S = (X, \rightarrow, \leq)$. For all configurations s, t, s' and for all $a \in \Sigma$, $w \in \Sigma^*$ such that $s < t$ and $s \xrightarrow{a} s' \xrightarrow{w} t$ there exists a configuration t' such that $t \xrightarrow{a} t'$ and $s' < t'$.

Remark. Every WSTS is branch WSTS because compatibility implies branch-compatibility and further, wqo implies branch-wqo. However, the converse result does not hold true. A counter-example to this is the transition system that we are going to describe in the next section in figure 2. It is branch WSTS but is not WSTS.

We see that there are some interesting decidability results for branch WSTS.

Definition 3.6 (FRT)**.** For any configuration $s \in \mathcal{S}$, where $\mathcal{S} = (S, \rightarrow, \leq)$ is an ordered transition system, $FRT(\mathcal{S}, s)$, the *Finite Reachability Tree* defined in [8] from s is a directed unordered tree where nodes are labeled by configurations of S. Nodes are either *dead* or *live*. The root node is a live node n_0, labeled by s_0. A dead node has no child node. A live node n, corresponding to configuration t has one child n' for each successor $t' \in Post_{\mathcal{S}}(t)$. If along the path from the root to some node n (corresponding to the configuration t), there exists a node n' (corresponding to the configuration t') such that $t' \leq t$, we say that n' subsumes n, and then n is a dead node. Otherwise, n is live.

Definition 3.7. If we expand the definition of an FRT, and also subsume nodes t, t' if $t' \rightarrow t$ and $t \leq t'$, the resulting tree is an antichain reachability tree (AT), defined in [2].The antichain tree is shorter than the finite reachability tree, since if a node is subsumed in the FRT, it is subsumed in the AT. Furthermore, if there is an infinitely long strictly decreasing chain, it is finite in the AT but not in the FRT.

Proposition 3.1. *A finitely branching transition system $\mathcal{S} = (S, \rightarrow, \leq)$ that is branch-wqo has a finite FRT. Hence, it has a finite AT.*

Proof. Let us assume that the FRT is infinite. Since the transition system is finitely branching, for the FRT to be infinite, there has to be an infinitely long branch with an infinite run. But since the system is branch-wqo, there exist two configurations s, s' such that $s \leq s'$ and $s \xrightarrow{*} s'$ in the given infinite run. In that case, the node s' is marked as dead, and the tree is not explored any further. Thus, there is a contradiction. Hence, the FRT, and as a result, the AT also, is finite. □

We look at two important properties for transition systems without potentially infinite states. They are as follows.

Problem (Termination)**.** A transition system $\mathcal{S}$ is said to be *terminating* at an initial configuration s_0 if it has no infinite run starting from s_0.

Problem (Boundedness)**.** A transition system $\mathcal{S}$ is said to be *bounded* at an initial configuration s_0 if the set of reachable states, i.e. $Post^*(s_0)$ is finite.

The following proof is the generalization of the proof of Proposition 4.10 in [8].

Proposition 3.2. *A branch WSTS, with strict compatibility and $\leq$ a partial ordering, is unbounded iff there exist two configurations in the finite reachability tree such that $s_1 \xrightarrow{*} s_2$ and $s_1 < s_2$.*

Proof. Let us assume the system is unbounded, i.e. $Post^*_{\mathcal{S}}(s_0)$ is infinite. Then, there are an infinite number of distinct configurations which are reachable from s_0. We first show that there exists a computation starting from s_0 without any loop, where all configurations are distinct. We consider the finitely branching tree of all prefixes of computations, and prune this tree by removing prefixes which contain a loop. Because any reachable configuration can be reached without a loop, the pruned tree still contains an infinite number of prefixes. By Konig's lemma, there exists an infinite computation with no loop. Any computation starting from s_0 has a finite prefix labeling a maximal path in the FRT. Hence, there must be a node s_2 which is subsumed by a node s_1 such that $s_1 \neq s_2$. Since we assumed $\leq$ to be a partial ordering, and $s_1 \leq s_2$, we have $s_1 < s_2$.

Conversely, let us assume that there exist two nodes s_1, s_2 in FRT such that $s_1 \xrightarrow{\sigma} s_2$ and $s_1 < s_2$ with $\sigma = a_1 a_2 ... a_n$ hence there exist $t_1, t_2, ..., t_n$ such that $s_1 \xrightarrow{a_1} t_1 \xrightarrow{a_2} t_2 \xrightarrow{a_3} ... \xrightarrow{a_n} t_n = s_2$.

Since the system is strictly branch compatible, there exist $u_1, u_2, ..., u_n$ such that $s_2 \xrightarrow{a_1} u_1 \xrightarrow{a_2} u_2 ... \xrightarrow{a_n} u_n$ with $t_1 < u_1$, $t_2 < u_2$ ... and $t_n < u_n$. By iterating this process, we construct an infinite sequence of configurations $(s_k)_{k \geq 0}$ such that for all $k \geq 1$ one have $s_k \xrightarrow{\sigma} s_{k+1}$ and $s_k < s_{k+1}$. Hence, $Post^*_{\mathcal{S}}(s_0)$ is infinite, and $\mathcal{S}$ is unbounded.

□

We may also generalize Proposition 4.5. for WSTS in [8] to branch WSTS, the proof is essentially the same as in [8].

Proposition 3.3. *Branch WSTS is non-terminating iff there exists a subsumed node in the FRT.*

Proof. Let us assume that the system is non-terminating, i.e. it has at least one infinite run. When we look at the prefix of the run on the FRT, since the FRT is finite, it ends in a dead node, say s_2, which means that there is a node s_1 in the run which subsumes s_2.

Conversely, if there is a subsumed node in the FRT, there is a non empty run and two configurations s_1, s_2 such that $s_1 \xrightarrow{\sigma} s_2$ and $s_1 \leq s_2$, where $\sigma = a_1 ... a_n$ and $\sigma \neq \epsilon$. Hence there exist $t_1, t_2, ..., t_n$ such that $s_1 \xrightarrow{a_1} t_1 \xrightarrow{a_2} t_2 \xrightarrow{a_3} ... \xrightarrow{a_n} t_n = s_2$. Since the system is strongly branch compatible, there exist $u_1, u_2, ..., u_n$ such that $s_2 \xrightarrow{a_1} u_1 \xrightarrow{a_2} u_2 ... \xrightarrow{a_n} u_n$ with $t_1 \leq u_1$, $t_2 \leq u_2$ ... and $t_n \leq u_n$. By iterating this process, we construct an infinite sequence of configurations $(s_k)_{k \geq 0}$ such that for all $k \geq 1$ one have $s_k \xrightarrow{\sigma} s_{k+1}$ and $s_k \leq s_{k+1}$. Hence, we have an infinite computation, and the system is non-terminating. □

The two above propositions give us the following result.

Theorem 3.1. *Termination is decidable for strict branch WSTS, with a decidable quasi-ordering $\leq$, and computable successor. Furthermore, if $\leq$ is a partial ordering with strict branch compatibility, boundedness is also decidable.*

Proof. We can construct the FRT for the branch WSTS since it is finite and we can compute the successors. If we find a node which is subsumed (by a strictly larger node for checking boundedness), then we can conclude non-termination (resp. unboundedness). This is possible since the ordering $\leq$ is decidable. □

Remark. For branch WSTS, we see that we cannot determine termination or boundedness using the AT like we can using the FRT. Look at the following counter-example 1.

If we consider the below FIFO machine (which is branch WSTS) , we see that it has an unbounded run of the form $(q_0, \epsilon) \xrightarrow{!a} (q_1, a) \xrightarrow{?a} (q_1, \epsilon) \xrightarrow{!b} (q_2, b) \xrightarrow{?b} (q_3, \epsilon) \xrightarrow{!b}$ .... This run in the AT will be as follows $n_0(s_0 = (q_0, \epsilon)) \rightarrow n_1(s_1 = (q_1, a)) \rightarrow n_2(s_2 = (q_1, \epsilon))$ and since $s_2 \leq s_1$, the node n_2 is subsumed by n_1 and marked dead. Hence, the automata has no branch in the AT such that there exists nodes labelled by s_1, s_2 where $s_1 \xrightarrow{*} s_2$ and $s_1 \leq s_2$ but it is unbounded.

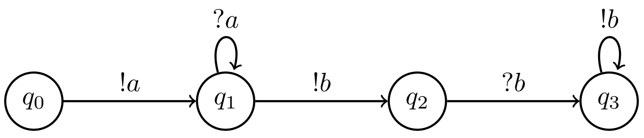


Figure 1: A FIFO machine with an unbounded run. $\mathcal{S}_1$

.

4 Bounded FIFO machines

Given three words $x, y, z \in \Sigma^*$ such that $z = x.y$, we say that x (resp. y) is a *left* (resp *right*) *factor* of z. We denote by $LF(z)$ and $RF(z)$ the sets of left and right factors of z. We say that a language $L \subseteq \Sigma^*$ is *bounded* if there exist $w_1, ..., w_n \in \Sigma^*$ such that $L \subseteq w_1^*...w_n^*$. Furthermore, a language $L \subseteq \Sigma^*$ is *letter-bounded* if there exist $v_1, ..., v_n \in \Sigma$ such that $L \subseteq v_1^*...v_n^*$, and more specifically, *different-letter-bounded* if for all $i, j \in \{1, ..., n\}$, $i \neq j \implies v_i \neq v_j$.

Remark. The set of prefixes of a bounded language is still a bounded language and that a finite union of bounded languages is still a bounded language: if $L \subseteq \bigcup_{i \in I} B_i$ where all B_i are bounded languages then $L \subseteq B$ where $B = \Pi_{i \in I} B_i$ is still a bounded language.

We first define a labelled transition system.

Definition 4.1 (Labeled transition system). A labeled transition system is a triple $TS = (Q, A, \rightarrow)$ where

- Q is the nonempty, countable set of states,
- A is the countable set of labels (or actions),
- $\rightarrow \subseteq Q \times A \times Q$ is the transition relation.

Given a transition $(q, a, q') \in \to$, q is called the source, q' the target and a the label of the transition. We assume that for any $a \in A$ there exists a transition $(q, a, q') \in \to$. A *rooted labeled transition system* is a pair (TS, q_0) where $TS = (Q, A, \to)$ is an LTS and $q_0 \in Q$ is the initial state (or root). Sometimes we write $TS = (Q, A, \to, q_0)$ for a rooted LTS.

We now describe a FIFO machines, which are the models that we consider in this section.

Definition 4.2. A FIFO machine $\mathcal{S}$ with one channel c over the alphabet Σ is an LTS defined as $\mathcal{S} = (Q, \Sigma, T)$, where Q is a finite set of control states, and $T \subseteq Q \times \{!, ?\} \times \Sigma \times Q$ is the transition relation. We denote by $A(\mathcal{S})$ the finite automaton associated with the FIFO machine $\mathcal{S}$.

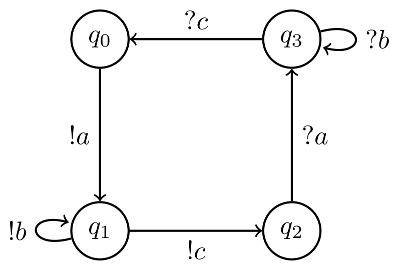


Figure 2: The FIFO machine $\mathcal{S}_1$

.

A *configuration* s is defined as the tuple $s = (q, w)$, where q is the control state of the machine, and w is the content of the queue at that point. Further, given a transition $t = (q, \#a, q')$, we understand it as $(q, w) \xrightarrow{\#a} (q', w')$, such that if $\#a = ?a$, for some letter a, $w = a.w'$, and if $\#a = !a$, $w.a = w'$. In other words, $?a$ represents the action of reading a letter a from the channel, and $!a$ represents writing the letter a to the end of the channel.

If we consider $\sigma = t_1 t_2 ... t_n$ to be a sequence of transitions from $s = (q, w)$ to $s' = (q', w')$, we have that $s \xrightarrow{\sigma} s'$ implies $\exists s_1, s_2, ..., s_n$ such that $q \xrightarrow{a_1} q_1 \xrightarrow{a_2} q_2 \xrightarrow{a_3} ... \xrightarrow{a_n} q_n = q'$, such that $(q, a_1, q_1), (q_1, a_2, q_2)... \in T$. If we consider $y_\sigma = \prod a_i$, such that $a_i = !a$, for some $a \in \Sigma$, and $x_\sigma = \prod a_j$ where $a_j = ?a$, for some $a \in \Sigma$, we have $w.y_\sigma = x_\sigma.w'$.

x_σ, short x is the word consumed by σ, and y_σ or y the word inserted by σ.

The *input language* $L_I(\mathcal{S}, s_0) = \{w \in \Sigma^* | w = proj(\sigma) \text{ and } s_0 \xrightarrow{\sigma} s\}$ of a FIFO machine (with a single channel c) is defined as the set of message sequences that fill up the channel. It is a language closed by left factors: $LF(L_I(\mathcal{S})) = L_I(\mathcal{S})$. The *reachability language* $L_R(\mathcal{S}, s_0)$ is defined as the set $\{w | (q, w)$ is reachable from s_0 for some $q \in Q\}$. In other words, the reachability language of the machine is defined as the set of all the words that are contained in the channel.

Definition 4.3. A FIFO machine $\mathcal{S} = (Q, \Sigma, T)$ is *input-bounded* in an initial state s_0 if its input language $L_I(\mathcal{S}, s_0)$ is bounded. $\mathcal{S}$ is *reachability-bounded* at s_0 if the language $Post^*_{\mathcal{S}}(s_0)$ is bounded. We say that $\mathcal{S}$ is *flat* if in $A(\mathcal{S})$, every state is in at most one unique elementary loop.

Remark. Input-bounded FIFO machines, and in fact one-letter FIFO machines, may simulate VASS. If the FIFO machine is input-bounded, it is also reachability-bounded, but the converse is not true, ref. to Example 2: the input language $(ab^*c)^*$ is not bounded while the reachability language of the fifo is included in the right factors of the language ab^*c. So we deduce that: VASS $\subsetneq$ input-bounded FIFO machine $\subsetneq$ reachability-bounded FIFO machine.

Remark. Flat FIFO systems are input-bounded but the converse is false: consider an unique control-state with two loops labeled by $!a$ and $!(aa)$. This system is not flat but its input-laguage is included in the bounded language a^* so this system is input-bounded.

Proposition 4.1. *The reachability set of a reachability-bounded FIFO machine is a bounded language.*

Proof. Consider a reachability-bounded FIFO machine $\mathcal{S}$, and an initial configuration s_0 such that the reachability language $L_R(\mathcal{S}, s_0)$ is $w_1^*...w_n^*$. Let w_0 be the contents of the channel at s_0. Then, we see that the reachability set of the FIFO machine is as follows:

$$Post_{\mathcal{S}}^*(s_0) \subseteq \bigcup_{k=1}^{n} \left[LF\Big(RF(w_0.w_k\Big) \cup \bigcup_{l=k}^{n} \left(RF(w_0.w_k. \prod_{h=k}^{l} w_h^*.LF(w_l) \right) \right].$$

Hence, it is a bounded language. □

4.1 Reachability-bounded FIFO machines simulate Minsky machines

While input-bounded FIFO systems have decidable boundedness and termination, for reachability-bounded systems, these problems are undecidable. This can be shown by representing a two-counter Minsky machine using a reachability-bounded system.

Proposition 4.2. *Two counter Minsky machines are simulated by reachability-bounded FIFO machines.*

Proof. Consider a Minsky machine $M = (\{q_0, ..., q_n\}, \{x_1, x_2\}, \{\delta_0, ..., \delta_{n-1}\})$, where q_i are the states, x_a and x_b are the counters, and δ_i are transition rules. The transition rules can be of two types -

$\delta_i : x_j := x_j + 1;$ `goto` q_k;

$\delta_i :$ `if` $x_j > 0$ `then` $(x_j := x_j - 1;$ `goto` $q_k)$ `else goto` q_l;

We can construct an equivalent reachability bounded machine as follows. $Q = \{q_1, ..., q_n\} \uplus Q'$, where Q' is a set of intermediate states. $\Sigma = \{a, b, \#, \$, \&\}$, where $\#, \$, \&$ are used as markers during the test for zero. We consider the bounded language to be $\$^*a^*\#^*b^*\&^*\$^*a^*\#^*b^*\&^*$, and we start with queue contents as $\$\#\&$. The intuition is to rotate the tape contents in such a way that the contents are always a bounded language, when incrementing or decrementing either counter. For every transition of the form $\delta_i : x_j := x_j + 1;$ `goto` q_k;, we create the following transition sequence that maintains the boundedness of the queue contents while incrementing the occurences of a and b depending on the original transition. (We assume that we can concatenate transitions to form single transitions, but this can be unraveled to ensure that every transition is only a single read or a single write.)

$$\begin{tikzcd}
 & q'_{i,2} \arrow[d, bend left, "!a"] & & & q'_{i,6} \arrow[d, bend left, "!b"] & \\
q_i \arrow[r, "?\$.!\$"] & q'_{i,1} \arrow[u, bend left, "?a"] \arrow[r, "?\#"] & q'_{i,3} \arrow[r, "!a"] & q'_{i,4} \arrow[r, "!\#"] & q'_{i,5} \arrow[u, bend left, "?b"] \arrow[r, "?\&.!\&"] & q_k
\end{tikzcd}$$

Figure 3: Incrementing x_a

.

Likewise, a transition of the form δ_i : `if` $x_j > 0$ `then` $(x_j := x_j - 1;$ `goto` $q_k)$ `else goto` q_l; can be constructed as follows.

$$\begin{tikzcd}
 & q'_{i,2} \arrow[d, bend left, "!a"] & & & q'_{i,6} \arrow[d, bend left, "!b"] & \\
q_i \arrow[r, "?a"] \arrow[d, "?\#.!\#"] & q'_{i,1} \arrow[u, bend left, "?a"] \arrow[r, "?\#"] & q'_{i,3} \arrow[r, "!a"] & q'_{i,4} \arrow[r, "!\#"] & q'_{i,5} \arrow[u, bend left, "?b"] \arrow[r, "?\&.!\&"] & q_k \\
q''_{i,1} \arrow[r, "?b!b"] & q''_{i,2} \arrow[r, "?\&.!\&"] & q_l & & &
\end{tikzcd}$$

Figure 4: Decrementing x_a

.

Similar constructions can be made for all transitions, and the queue contents are always of the form $\$^* a^* \#^* b^* \&^* \$^* a^* \#^* b^* \&^*$, hence, the system is reachability-bounded.

$\square$

4.2 Prefix ordering over bounded automata

The (extended) prefix ordering on the configurations of a FIFO machine is defined as follows: $s = (q, w) \leq_{pref} s' = (q', w') \implies q = q'$ and $w \preceq w'$. The extended prefix-ordering is reflexive, antisymmetric, and transitive, hence, it is an ordering.

Proposition 4.3. *Input-bounded FIFO machines are branch-wqo for the prefix-ordering $\leq_{pref}$.*

Proof. Let us consider a FIFO system $\mathcal{S}$ and an infinite run $r = s_1 s_2 ... s_i ...$ with $s_i = (q_i, w_i)$. Since $\mathcal{S}$ is input-bounded, the reachability set $Post^*_{\mathcal{S}}(s_0)$ is also a bounded language (Proposition 4.1) included in $v_1^* v_2^* ... v_k^*$ where all $v_i \in \Sigma^*$.

The infinite run is of the form $s_0 \xrightarrow{\sigma_1} s_1 \xrightarrow{\sigma_2} s_2 ... s_{i-1} \xrightarrow{\sigma_i} s_i$ and we denote $\sigma = \sigma_1 \sigma_2 ... \sigma_i ...$ and $\sigma[i] = \sigma_1|_! \sigma_2|_! ... \sigma_i|_!$. It can be observed that $\sigma[i]$ is a prefix of $\sigma[i+1]$ for all $i \in \mathbb{N}$. Since $\sigma[i]$ is of the form $v_1^{n_{1,i}} ... v_m^{n_{m,i}} LF(v_m)$ for some $n_{1,i} ..., n_{m,i} \geq 0$ and $1 \leq m \leq k$, the infinite sequence $(\sigma[i])_{i \in \mathbb{N}}$ satisfies two possible exclusive cases:

case (1) : There exists an i_0 such that $\forall i \geq i_0, \sigma_i|_! = \epsilon$ so there exists $i_1 \geq i_0$ such that for all $i \geq i_1, w_i = w_{i+1}$. Hence because there are finitely many control states, we deduce that there exist $i_2, i_3 \geq i_1$ st $s_{i_2} \xrightarrow{+} s_{i_3}$ and $s_{i_2} = s_{i_3}$ hence also in particular $s_{i_2} \leq_{pref} s_{i_3}$.

case (2) : There are infinitely many indices i such that $\sigma_i|_! \neq \epsilon$, which means that the infinite sequence $(\sigma[i])_{i \in \mathbb{N}}$ is not stationary. This implies that the set $S_\sigma = \{(n_{1,i}, ..., n_{k,i}) \mid i \in \mathbb{N}\}$, associated with σ, is infinite. Hence, there exists a least index p such that the set $\{n_{p,i}\}_{i \in \mathbb{N}}$ is infinite. Then the set $F = \{(n_{1,i}, ..., n_{p-1,i}) \mid i \in \mathbb{N}\}$ is finite.

We claim that for all indices $\ell \geq p+1$, $n_{\ell,i} = 0$ for all i. Let us assume to the contrary that there is some index $\ell \geq p+1$ and i_0 such that $n_{\ell,i_0} \neq 0$. This means that the word v_ℓ is in the channel in configuration s_{i_0}, which means that the word v_ℓ was sent to the channel before (or at) the step i_0, i.e, $\sigma[i_0] = v_1^{n_{1,i_0}} ... v_p^{n_{p,i_0}} ... v_m^{n_{m,i_0}} LF(v_m)$ and $n_{l,i_0} > 0$ and $1 \leq m \leq k$. So, in particular, word v_p cannot be sent after i_0, hence, $n_{p,i} = n_{p,i_0}$ $\forall i > i_0$. Hence, $\{n_{p,i}\}_{i\in\mathbb{N}}$ is finite which is a contradiction to our assumption that $\{n_{p,i}\}_{i\in\mathbb{N}}$ is infinite.

This means that after some configuration s_t, we only write word v_p to the channel. Since, the set $F = \{(n_{1,j}, ..., n_{p-1,j}) \mid j \in \mathbb{N}\}$ is finite, we can extract an infinite subsequence $(q, w_i)_{i\in K\subseteq\mathbb{N}}$ where $w_i = uv_p^{n_{p,i}}$ with $u \in F$ and $(n_{p,i})_{i\in K}$ is non-decreasing. Hence, there exist two indices $a, b > 0$ such that $w_a = u.v_p^{n_{p,a}}$ and $w_{a+b} = u.v_p^{n_{p,a+b}}$ and $n_{p,a} \leq n_{p,a+b}$ hence $w_{a+b} = w_a.v_p^{n_{p,a+b}-n_{p,a}}$ hence $w_a \leq_{pref} w_{a+b}$. So we have found two configurations s_a, s_{a+b} such that $s_a \leq_{pref} s_{a+b}$. Hence, the system is branch-wqo for the prefix ordering.

□

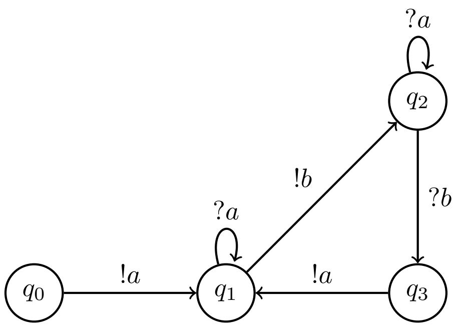


Figure 5: The FIFO machine $\mathcal{S}_1$

.

Remark. System S_1 in fig 5 shows that input-bounded FIFO machine under the prefix-ordering are not branch-compatible. We have $(q_1, \epsilon) \leq_{pref} (q_1, a)$ and $(q_1, \epsilon) \xrightarrow{!b?b!a} (q_1, a)$. However, $(q_1, \epsilon) \xrightarrow{!b} (q_2, b)$ and $(q_1, a) \xrightarrow{!b} (q_2, ab)$, but $b \not\leq_{pref} ab$. Therefore, the system is not branch-compatible under the prefix ordering.

Proposition 4.4. *Input-bounded FIFO machines are not branch-compatible for the prefix-ordering.*

4.3 Prefix-compatible relation over bounded machines

We first define a new relation which is a restriction of the prefix ordering, and then see that bounded FIFO machines are branch WSTS over this relation.

For any sequence σ of transitions in a FIFO machine, we denote by x_σ the concatenation of all the letters received from the channel in σ, and y_σ is the concatenation of all the letters sent to the channel in σ.

Definition 4.4. For two configurations $(q, w), (q', w')$ of a FIFO machine, we say that $(q, w) \leq_{comp} (q', w')$ if $(q, w) \xrightarrow{\sigma} (q', w')$ and ($x_\sigma = \epsilon$ and $q = q'$) or the three following conditions hold:

- $(q, w) \leq_{pref} (q', w')$ and
- $|x_\sigma| \leq |y_\sigma|$ and
- $x_\sigma^\omega = w.y_\sigma^\omega$.

Intuitively, this relation implies that σ is infinitely iterable from the configuration (q, w), as shown as part of Lemma 3.8 in [11]. The following lemma from [11] and [9] is used in order to prove it.

Proposition 4.5. *Consider three finite words $x, y \in \Sigma^+$ and $w \in \Sigma^*$. The equation $x_\sigma^\omega = w.y_\sigma^\omega$ holds iff there exists a primitive word $z \neq \epsilon$ and two words x', x'' such that $x = x'x''$, $x''x' = z^*$, $w = x^*x'$ and $y = z^*$.*

We assume that $x \neq \epsilon$. That implies $x_\sigma^\omega = w.y_\sigma^\omega$. We infer from the previous proposition that there is a primitive word $z \neq \epsilon$ and two words x', x'' such that $x = x'x''$, $x''x' = z^*$, $w = x^*x'$ and $y = z^*$ Suppose $x''x' = z^j$ and $y = z^k$. Since $|y| \geq |x| = |x''x'|$, we have $k \geq j$.

Proposition 4.6 ([11])**.** *For all $n \geq 0$, σ is fireable from any channel content wz^n and the resulting channel contains $wz^{n+(k-j)}$. This implies that for all $m \geq 1$, $(q, w) \xrightarrow{\sigma^m} (q, wz^{m\times(k-j)})$, hence, σ is infinitely iterable.*

Proposition 4.7. *FIFO machines are branch-compatible for the relation $\leq_{comp}$.*

Proof. In other words, we consider a FIFO machine with configurations s, t, s' such that $s \leq_{comp} s'$ with $s \xrightarrow{a_1} t \xrightarrow{a_2...a_n} s'$ and let us write $\sigma = a_1a_2...a_n$. Then we need to show that there exists a configuration t' such that $s' \xrightarrow{a_1} t'$ and $t \leq_{comp} t'$ where $\sigma' = a_2...a_na_1$.

There are two cases, either that a_1 is a *send* transition, or it is a *receive* transition. For both cases, we know that $s \leq_{comp} s'$, hence $q(s) = q(s')$. Therefore, the transition a_1 is enabled at s', and further, there exists a configuration t' such that $s' \xrightarrow{a_1} t'$ such that $q(t) = q(t')$. Now we have to show that $t \leq_{comp} t'$.

- (Case 1 : *receive*) Let us assume $a_1 = ?a$ for some $a \in \Sigma$. Since, a_1 is a receive transition, $w(s) = a.w(t)$. Further, $y_\sigma = y_{\sigma'}$, since all the write transitions are the same for σ and σ'. Also, $a.x_{\sigma'} = x_\sigma.a$. Hence, $x_\sigma = a.p$ and $x_{\sigma'} = p.a$ for some w.

  Since $s \leq_{comp} s'$,
  $$\begin{aligned} & w(s).y_\sigma^\omega = x_\sigma^\omega \\ \implies & a.w(t).y_{\sigma'}^\omega = (a.p)^\omega \\ \implies & a.w(t).y_{\sigma'}^\omega = a.(p.a)^\omega \\ \implies & w(t).y_{\sigma'}^\omega = x_{\sigma'}^\omega \end{aligned}$$

  Hence, $t \leq_{comp} t'$.

- (Case 2 : *send*) Let us assume $a_1 = !a$ for some $a \in \Sigma$.

  Since, a_1 is a send operation, $w(s).a = w(t)$. Further, $x_\sigma = x_{\sigma'}$, since all the letters received are the same for σ and σ'. Also, $a.y_{\sigma'} = y_\sigma.a$. Hence, $y_\sigma = a.p$ and $y_{\sigma'} = p.a$ for some w. Since $s \leq_{comp} s'$,
  $$\begin{aligned} & w(s).y_\sigma^\omega = x_\sigma^\omega \\ \implies & w(s).(a.p)^\omega = x_{\sigma'}^\omega \\ \implies & w(s).a.(p.a)^\omega = x_{\sigma'}^\omega \\ \implies & w(t).y_{\sigma'}^\omega = x_{\sigma'}^\omega \end{aligned}$$

  Hence, $t \leq_{comp} t'$.

Therefore, FIFO machines are branch-compatible for the relation $\leq_{comp}$ □

Remark. FIFO machines under the relation $\leq_{comp}$ are also strictly branch-compatible. Let us consider two configurations s, s' such that $s <_{comp} s'$ for the word $\sigma = a_1a_2...a_n$. Then if $s \xrightarrow{a_1} t$ and a_1 is a read operation, then the resulting t' such that $s' \xrightarrow{a_1} t'$ gives us the relation $t <_{comp} t'$, since the $w(s) \neq w(s') \implies a.w(t) \neq a.w(t') \implies w(t) \neq w(t')$. Now, if a_1 is a write action, then $w(s) \neq w(s') \implies w(s).a \neq w(s').a \implies w(t) \neq w(t')$. Hence, FIFO machines are *strictly branch-compatible* under this relation.

However, since the relation $\leq_{comp}$ is not an ordering, we cannot use the theory of branch-wqo to it. But in [13], the following result was proved.

Proposition 4.8. *[13]The FRT is finite for the relation $\leq_{comp}$ for input-bounded FIFO systems.*

Using the result in Proposition 3.3 and the above proposition along with the fact that the relation $\leq_{comp}$ is branch-compatible for FIFO machines, we have the proposition.

Corollary 4.0.1. *Termination is decidable for input-bounded FIFO machines.*

Decidability of boundedness of input-bounded FIFO machines was proved earlier in [13]. Hence, combining the two propositions, we have the following result.

Theorem 4.1. *Boundedness and termination are decidable for input-bounded FIFO machines.*

5 Reachability of input-bounded FIFO machines

We now look at the below two verification properties for transition systems.

Problem (Reachability)**.** Given a system $\mathcal{S}$ and two configurations (q_0, w_0) and (q, w), is there a run starting from (q_0, w_0) which reaches (q, w)?

Problem (Control-state reachability)**.** Given a FIFO system $\mathcal{S}$, a configuration (q_0, w_0) and a control-state q, is there a channel valuation w such that (q, w) is reachable from (q_0, w_0)

In [11], it was shown that reachability reduces to control-state reachability for flat systems. However, in case of flat systems, the converse is not true. However, using the same reductions as in [11], we have the following result for input-bounded FIFO machines.

Proposition 5.1. *Reachability is recursively equivalent to control-state reachability for input-bounded FIFO systems.*

Proof. Let us consider a FIFO machine $\mathcal{S}$, a control-state q, and a configuration (q, w). We first reduce reachability to control state reachability. Another system $\mathcal{B}_{\mathcal{S},(q,w)}$ is constructed as follows. A path is added from the control state q as follows, where \$ is a new symbol not in Σ.

$$\begin{tikzcd} q \arrow[r, "!\$"] & \bullet \arrow[r, "?w\$"] & q_{\text{end}} \end{tikzcd}$$

The configuration (q, w) is reachable in $\mathcal{S}$ iff the control state q_{end} is reachable in $\mathcal{B}_{\mathcal{S},(q,w)}$. Furthermore, $\mathcal{B}_{\mathcal{S},(q,w)}$ is input-bounded if the initial system is input-bounded, since we only append a finite word to the existing system, and concatenation of a bounded-language with a finite string results in a bounded language. Therefore, reachability reduces to control-state reachability for input-bounded FIFO systems.

Conversely, in order to show that control-state reachability is reducible to reachability, we construct $\mathcal{B}_{\mathcal{S},q}$ as follows. To $\mathcal{S}$, we add $|\Sigma|$ self-loops from and to the control state q, each labeled by a unique letter in the alphabet Σ. q is reachable in $\mathcal{S}$ iff there exists a w such that (q, w) is reachable in $\mathcal{S}$ iff (q, ϵ) is reachable in $\mathcal{B}_{\mathcal{S},q}$. We see that adding self-loops with read transitions does not change the input-language of the system, and hence, if $\mathcal{S}$ is input-bounded, so is $\mathcal{B}_{\mathcal{S},q}$. Hence, control state reachability reduces to reachability for input-bounded FIFO systems. □

Remark. Even though we can extend the established results to show that reachability and control-state reachability are reducible to one another, we don't see how to extend the reachability result shown in [7]. This is because the proof uses a *d-tape pushdown automata* in order to show that reachability is in NP. However, the trace of the input-bounded FIFO machines need not be a bounded language. Hence, it is still an open problem to know the complexity of the reachability of input-bounded FIFO systems.

5.1 Reachability of input-letter bounded FIFO systems

In [5], it was shown that reachability is decidable in input-*different-letter* bounded FIFO systems. The proof idea involved simulating such systems using Petri nets with a structured set of terminal markings. We show that reachability in input-*letter* bounded is also decidable.

(*Intuition*) In order to show reachability is decidable for *input-letter-bounded* FIFO systems, we first construct a counter machine. Let us assume that $S = (Q, \Sigma, T)$ is a FIFO system over a channel c, and the input language of the system is $v_1^* ... v_k^*$. We construct the corresponding counter machine $S' = (Q', K, \Delta)$.

We consider three automata, which share counters and synchronize them. The counters shared between the automata are as follows: there is one counter c_α and another counter c'_α, for each $\alpha \in \Sigma$. The counter c_α is incremented when there is a send operation, and the counter c'_α is decremented when there is a receive operation. Furthermore, we have k additional counters for each of the letters $v_1, ..., v_k$ in the input language.

The first automaton has the same set of states as the original FIFO machine, and transitions mimicking the one in the original machine. However, instead of sending and receiving from the channel, the transition modifies the counter corresponding to the letter being sent/received.

The second automaton is to ensure that the input language of the machine is letter bounded. Since we know the input-language of the machine, we can ensure that the input language of the counter automaton is the same as the FIFO machine. It has states $\{send_i\}$ for all i from 1 to k, and each state increments the counter corresponding to v_i and ensures that no preceding letter v_j can be sent to the channel if we are already receiving v_i where $i > j$.

The third automaton ensures that reception happens in the same order as the send, and that any letter can be read only after it is sent, and all previously sent letters must have already been received. It has states $\{rec_i\}$ for all i from 1 to k. But since we can't receive letters which have not been sent, the product ensures that the counter machine cannot be in a state $(q, send_i, rec_j)$ such that $i < j$.

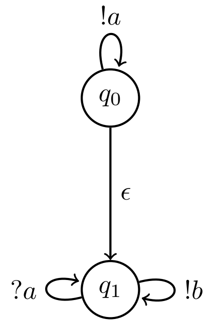
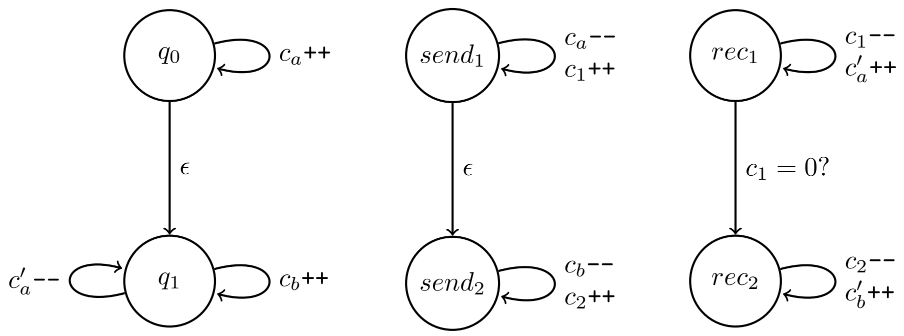


Figure 6: A FIFO machine $\mathcal{S}$ with input-language a^*b^*.



$\boxed{\text{Constraints: } \sum_{\alpha\in\Sigma} c'_\alpha \leq 1 \text{ and } \sum_{\alpha\in\Sigma} c_\alpha \leq 1}$

Figure 7: Counter automata corresponding to the FIFO machine $\mathcal{S}$.

$Q' = Q \times \{send_i, rec_j | i \in \{1, ..., k\}$ and $j \in \{1, ..., i\}\}$.

$K = \{c_\alpha, c'_\alpha | \alpha \in \Sigma\} \bigcup \{c_i | i \in \{1, ..., k\}\}$.

The counters are as follows : c_a is incremented when a is sent to the channel, c'_a is decremented when a is received from the channel, and c_i keeps track of the number of occurences of letter v_i. We also have the following constraints-

$$\sum_{\alpha\in\Sigma} c'_\alpha \leq 1$$

$$\sum_{\alpha\in\Sigma} c_\alpha \leq 1$$

These constraints ensure that only one of the counters in the sets c_α and c'_α can be non-zero, we cannot store multiple occurences of letters in them, and instead, we store them in the counters $c_1, ..., c_k$.

$\Delta = \Delta_S \bigcup \Delta_{read} \bigcup \Delta_{write}$, where Δ_S mimics the FIFO system, i.e.
if $q \xrightarrow{!a} q' \in T$ then $(q, send_i, rec_i) \xrightarrow{c_a{+}{+}} (q', send_i, rec_j) \in \Delta_S$
if $q \xrightarrow{?a} q' \in T$ then $(q, send_i, rec_j) \xrightarrow{c'_a{-}{-}} (q', send_i, rec_j) \in \Delta_S$

For every letter in the alphabet Σ and the input language, if $v_i = a$, we add the following transitions to ensure that we read the letters. We count the occurences of the send operations for each letter v_i and increment the counter c_i. Furthermore, since we know that the input language is bounded, when we increment a counter c_i, we cannot ever send a letter v_j such that $j < i$ again, hence, all the counters c_j will never be incremented again.
$(q, send_i, rec_j) \xrightarrow{c_a{-}{-};c_i{+}{+}} (q, send_i, rec_j) \in \Delta_{write}$ and
$(q, send_i, rec_j) \xrightarrow{\quad\epsilon\quad} (q, send_{i+1}, rec_j) \in \Delta_{write}$ for all $1 \leq j \leq i < k$.

And for the receive transitions, we just need to make sure we receive the letters in the same order that they were sent (FIFO behaviour). Hence, unless all the occurences of v_j are received and the counter is tested to zero before we can start receiving v_{j+1}.

$(q, send_i, rec_j) \xrightarrow{c'_a++; c_j--} (q, send_i, rec_j) \in \Delta_{read}$ and
$(q, send_i, rec_j) \xrightarrow{c_j=0?} (q, send_i, rec_{j+1}) \in \Delta_{read}$ for all $1 \leq j \leq i < k$.

The product looks as follows ($send_i$ and rec_i have been abbreviated to s_i and r_i respectively).

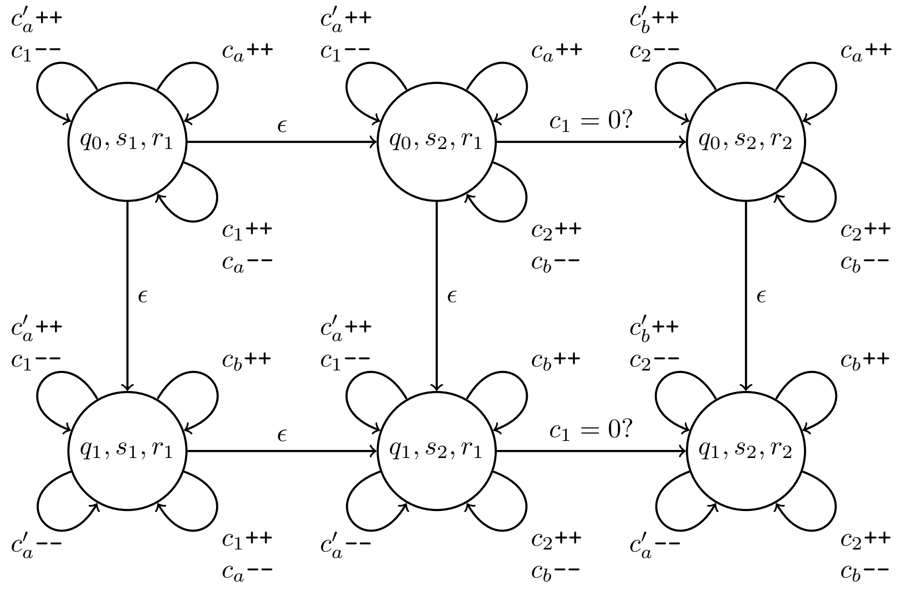


Figure 8: Synchronized product of counters simulating $\mathcal{S}$

Proposition 5.2. *If* $(q_0, \epsilon) \xrightarrow{\sigma} (q, w)$ *then there exists* $(q, send_i, rec_j)$ *reachable in* S' *such that the following conditions hold*

- $\sum_{\alpha \in \Sigma} c_\alpha = 0$
- $\sum_{\alpha \in \Sigma} c'_\alpha = 0$
- $i \geq j$
- $\sum_{p=1}^{j-1} c_p = 0$
- $\sum_{p=i+1}^{k} c_p = 0.$

Furthermore, $\sigma|_! = v_1^{n_1}...v_i^{n_i}$ *where* $n_i > 0$ *if* $i > 1$ *and* $w = v_j^{m_j}...v_i^{m_i}$ *and* $c_\ell = m_\ell$ *for all* $j \leq \ell \leq i$.

Proof. Let us prove this by induction. The initial state q_0 is reachable in S and since $(q_0, send_1, rec_1)$ is the initial state in S' and all counters are set to zero, this holds true trivially.

Let us assume we can reach a state q in S such that $(q_0, \epsilon) \xrightarrow{\sigma} (q, w)$, and equivalently we can reach $(q, send_i, rec_j)$ such that the conditions are met. Now, we need to prove that any state reachable from q can be also reached in S'.

Let us assume we have a send operation such that $(q, w) \xrightarrow{!a} (q_1, w.a)$. Since we know that the input language is bounded, this corresponds to the fact that we are sending a letter

v_p where $p \geq i$. Since the letter counters are all set to zero, we can execute the transition $(q, send_i, rec_j) \xrightarrow{c_a++} (q_1, send_i, rec_j)$. Furthermore, since $p \geq i$, we can perform a sequence of empty transitions from Δ_{write} such that $(q, send_i, rec_j) \xrightarrow{\epsilon} (q, send_p, rec_j$ and then one final transition $(q, send_p, rec_j) \xrightarrow{c_a--;c_p++} (q, send_p, rec_j)$. This new configuration satisfies all our properties.

Let us assume that instead there is a receive operation. Since there is a read operation, it implies that at some point there was a sending to channel with the same letter, and all other letters before it have been consumed. Since the queue contents are of the form $w = v_j^{m_j}...v_i^{m_i}$, we have that the next letter to be received is v_j or if there is no v_j in the channel, some letter v_p such that $p > j$. If the current letter to be read is v_j, we can execute the following transitions on S'. $(q, send_i, rec_j) \xrightarrow{c'_a++;c_j--} (q, send_i, rec_j$ and then the transition $(q, send_i, rec_j) \xrightarrow{c'_a--} (q', send_i, rec_j)$. Otherwise, if there is no v_j in the channel, and the next occurence is some v_p and all counters $c_\ell = 0$ where $j \leq \ell \leq p$. Hence, we can execute the transitions $(q, send_i, rec_j) \xrightarrow{c_\ell=0?} (q, send_i, rec_p$, and then since C_p is non-zero, we can execute $(q, send_i, rec_p) \xrightarrow{c'_a++;c_p--} (q, send_i, rec_p$ followed by $(q, send_i, rec_p) \xrightarrow{c'_a--} (q', send_i, rec_p)$. $\square$

Proposition 5.3. *Conversely, if a configuration $(q, send_i, rec_j)$ is reachable in S' from the initial configuration, such that it satisfies the conditions in the previous proposition, i.e.*

- $\sum_{\alpha\in\Sigma} c_\alpha = 0$
- $\sum_{\alpha\in\Sigma} c'_\alpha = 0$
- $i \geq j$
- $\sum_{p=1}^{j-1} c_p = 0$
- $\sum_{p=i+1}^{k} c_p = 0$

then the control state q is reachable in S. Furthermore, the contents of the queue at q is $w = v_j^{m_j}...v_i^{m_i}$ where $m_\ell = c_\ell \neq 0$ (the current counter values) for all $j \leq \ell \leq i$.

Proof. We prove this by induction. The initial configuration in S' is $(q_0, send_1, rec_1)$, with all counters at 0. Since (q_0, ϵ) is the initial configuration in S, the proposition is trivially true.

Now, let us assume we can reach a configuration $(q, send_i, rec_j)$ in S' such that the conditions are met, and the control state q is reachable in S. We now show that if the control state q_1 is reachable in S', it is also reachable in S.

At the current configuration $(q, send_i, rec_j)$, we have the following options:

- Case 1 : $(q, send_i, rec_j) \xrightarrow{\epsilon} (q, send_p, rec_j)$ where $p > i$. In this case, since the control state does not change, and neither do the counter values, the result holds.
- Case 2: $(q, send_i, rec_j) \xrightarrow{\epsilon} (q_1, send_i, rec_j)$. This implies that there is an epsilon transition from q to q_1 in S, and hence, the result still holds.
- Case 3: $(q, send_i, rec_j) \xrightarrow{c_a++} (q_1, send_i, rec_j)$. But since the counter values do not satisfy the conditions (i.e. we are in an intermediate configuration), we then have $(q, send_i, rec_j) \xrightarrow{\epsilon} (q_1, send_p, rec_j) \xrightarrow{c_a--,c_p++}$, where $p \geq i$. Hence, the word $v_j^{m_j}...v_i^{m_i}v_p$ is input-bounded, and furthermore, the transition from q to q_1 marked with $!a$ can be executed in order to obtain the word $v_j^{m_j}...v_i^{m_i}v_p$. Hence, the result holds.

- Case 4: $(q, send_i, rec_j) \xrightarrow{c_j--; c'_a++} (q, send_i, rec_j)$. However, this is an intermediate configuration, hence, we then execute $(q, send_i, rec_j) \xrightarrow{c'_a--} (q_1, send_i, rec_j)$. Since the current content of the channel is $v_j^{m_j}...v_i^{m_i}$, it can be seen that v_j can be received from the queue in S, and the transition labeled with $?a$ from q to q_1 is feasible. Furthermore, if m_j then becomes zero, the transition $(q_1, send_i, rec_j) \xrightarrow{c_j=0?} (q_1, send_i, rec_{j+1})$ is executed.

Hence, for each of the possible valid transition sequences in S', we see that the corresponding control states are reachable in S. Hence, the result holds true. □

The above two propositions give us the following result.

Theorem 5.1. *Reachability is decidable for input-letter-bounded FIFO systems.*

6 Conclusion

We were able to show that termination and boundedness are decidable for input-bounded FIFO systems. Furthermore, we also showed that reachability is decidable for input-letter-bounded FIFO systems. We conjecture that it is also decidable for input-bounded FIFO systems, by modifying the counter systems to accept words instead of single letters. However, this is still an open question. While we have shown that termination, boundedness (and reachability for input-letter bounded systems) are decidable, the complexity of such problems is still an open question.

The approach of understanding FIFO systems using over and under-approximations is a interesting way forward. We have seen that there are subclasses which have decidable properties, and verification of general FIFO systems may be possible by using these approximations in a semi-decidable algorithm, if their complexity is efficient. Such methods could be implemented to partially analyse the system.

It would also be interesting to explore the notions of branch WSTS that we have defined in this report. And finally, it is not yet known whether the problem of coverability is decidable for input-bounded FIFO systems.

References

[1] Parosh Aziz Abdulla et al. "Using forward reachability analysis for verification of lossy channel systems". In: *Formal Methods in System Design* 25.1 (2004), pp. 39–65.

[2] Michael Blondin, Alain Finkel, and Pierre McKenzie. "Well Behaved Transition Systems". In: *Logical Methods in Computer Science* 13 (2016).

[3] Daniel Brand and Pitro Zafiropulo. "On communicating finite-state machines". In: *Journal of the ACM (JACM)* 30.2 (1983), pp. 323–342.

[4] Gérard Cécé and Alain Finkel. "Verification of programs with half-duplex communication". In: *Information and Computation* 202.2 (2005), pp. 166–190.

[5] A Choquet and A Finkel. "Simulation of linear FIFO nets having a structured set of terminal markings". In: *Proc. 8th European Workshop on Applications and Theory of Petri Nets.* 1987.

[6] P. Erdős and R. Rado. "A partition calculus in set theory". In: *Bull. Amer. Math. Soc.* 5 (Sept. 1956), pp. 427–489.

[7] 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. New Orleans, Louisiana: IEEE Computer Society, 2012, pp. 285–294.

[8] A. Finkel and Ph. Schnoebelen. "Well-structured transition systems everywhere!" In: *Theoretical Computer Science* 256.1 (2001), pp. 63–92.

[9] Alain Finkel, S. Purushothaman Iyer, and Grégoire Sutre. "Well-abstracted transition systems: application to FIFO automata". In: *Information and Computation* 181.1 (2003), pp. 1–31.

[10] Alain Finkel and Etienne Lozes. "Synchronizability of communicating finite state machines is not decidable". In: *ICALP* (2017).

[11] Alain Finkel and M. Praveen. "Verification of Flat FIFO Systems". In: *Proceedings of the 30th International Conference on Concurrency Theory (CONCUR'19).* Ed. by Wan Fokkink and Rob van Glabbeek. Leibniz International Proceedings in Informatics. To appear. Amsterdam, The Netherlands: Leibniz-Zentrum für Informatik, Aug. 2019.

[12] Blaise Genest, Dietrich Kuske, and Anca Muscholl. "On communicating automata with bounded channels". In: *Fundamenta Informaticae* 80.1-3 (2007), pp. 147–167.

[13] Thierry Jéron and Claude Jard. "Testing for Unboundedness of FIFO Channels." In: *Theor. Comput. Sci.* 113 (May 1993), pp. 93–117.

[14] Peizun Liu, Thomas Wahl, and Akash LaL. "Verifying Asynchronous Event-Driven Programs Using Partial Abstract Transformers (Extended Manuscript)". In: *arXiv preprint arXiv:1905.09996* (2019).