On Verifying Fair Lossy Channel Systems

B. Masson* and Ph. Schnoebelen

Lab. Spécification & Vérification
ENS de Cachan & CNRS UMR 8643
61, av. Pdt. Wilson, 94235 Cachan Cedex France
email: phs@lsv.ens-cachan.fr

Abstract. Lossy channel systems are systems of finite state automata that communicate via unreliable unbounded fifo channels. They are an important computational model because of the role they play in the algorithmic verification of communication protocols.
In this paper, we show that fair termination is decidable for a large class of these systems.

1 Introduction

Channel Systems are systems of finite state automata that communicate via asynchronous unbounded fifo channels (see example on Fig. 1). They are a natural model for asynchronous communication protocols and constitute the semantical basis for ISO protocol specification languages such as SDL and Estelle.

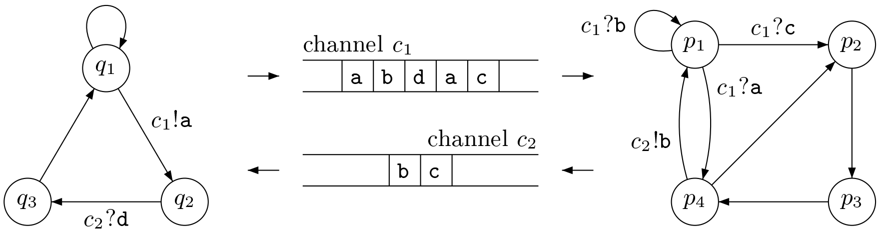

Fig. 1. A channel system with two component automata and two channels

Automated verification of channel systems. Formal verification of channel systems is important since even the simplest communication protocols can have tricky behaviors and hard-to-find bugs. But channel systems are Turing powerful [1], and no verification method for them can be general and fully algorithmic.

* Now at Dept. Comp. Sci., ENS de Lyon. Email: bmasson@ens-lyon.fr. The research described in this paper was conducted while B. Masson was at LSV.

[1] A Turing machine is easily simulated (with polynomial-time overhead) by a single-channel system that stores in its channel the contents of the Turing machine work tape plus a marker for the current position of the reading head [BZ81].

Lossy channels. A few years ago, Abdulla and Jonsson identified *lossy channel systems* as a very interesting model: in lossy channel systems messages can be lost while they are in transit, without any notification [2]. These lossy systems are the natural model for fault-tolerant protocols where the communication channels are not supposed to be reliable (see [ABJ98,AAB99] for applications). Surprisingly, some verification problems become decidable when one assumes channels are lossy: reachability, safety properties over traces, and inevitability properties over states are decidable for lossy channel systems [Fin94,AK95,CFP96,AJ96b].

One should not believe that lossy channel systems are trivial models where no interesting behavior can be enforced (since messages can always be lost), so that most verification problems would be vacuously decidable. Quite the opposite is true, and many problems are undecidable for these systems: recurrent reachability properties are undecidable, so that temporal logic model-checking is undecidable too [AJ96a]. Furthermore, boundedness is undecidable [May00], as well as all behavioral equivalences [Sch01]. Finally, all the known decidable problems have nonprimitive recursive complexity [Sch02] and are thus much harder than most decidable verification problems.

Fairness properties. The most important undecidable problem for lossy channel system is *recurrent control state reachability* (RCS), shown undecidable by Abdulla and Johnson [AJ96a]. RCS asks whether there exists a run visiting a given control state infinitely often (i.e. an infinite run satisfying a Büchi acceptance condition). The undecidability of RCS is often summarized by the slogan "*fairness properties are undecidable for lossy channel systems*".

Our contribution. In this paper we show that, in fact, there exist natural fairness properties that are decidable for lossy channel systems. Indeed, we show that termination under the assumption of fair scheduling ("fair termination") is decidable for a large and natural class of lossy channel systems: those where the channels are not used to multiplex messages aimed at different components. The underlying reason is that, for such systems, termination is "insensitive to fairness". This positive result applies to weak and strong fairness equally.

A second, more surprising and technically more involved result, is that termination for weakly fair scheduling is decidable for single-channel systems.

These two positive results are close to the frontier of decidability: we show that undecidability appears after a slight weakening of the hypothesis. Furthermore, for strongly fair termination, we precisely characterize the communication layouts that ensure decidability, showing that multiplexed channels really are the central issue.

Finally, beyond termination, there is only one other decidable problem that can meaningfully be investigated under the assumption of fair scheduling, namely *inevitability properties*. We show that these properties immediately become undecidable when fair scheduling is assumed.

[2] These systems are very close to the *completely specified protocols* independently introduced by Finkel [Fin94].

Plan of the paper. We first recall the necessary notions in Section 2. Then we study fair termination for systems without multiplexed channels in Section 3, and for single-channel systems in Section 4. Characterization of the layouts ensuring decidability is done in Section 5. Finally, Section 6 discusses fair inevitability.

2 Channel systems

Given a finite alphabet $\Sigma = \{a, b, \ldots\}$, we let $\Sigma^* = \{u, v, \ldots\}$ denote the set of all finite words over Σ. For $u, v \in \Sigma^*$, we write $u.v$ (also uv) for the *concatenation* of u and v. We write ε for the empty word and Σ^+ for $\Sigma^* \setminus \{\varepsilon\}$. The length of $u \in \Sigma^*$ is denoted $|u|$.

The *subword relation*, denoted $u \sqsubseteq v$, relates any two words u and v s.t. u can be obtained by erasing some (possibly zero) letters from v. For example $\mathtt{abba} \sqsubseteq \underline{\mathtt{ab}}\mathtt{racada}\underline{\mathtt{b}}\mathtt{r}\underline{\mathtt{a}}$, (the underlined letters are not erased). We write $u \sqsubset v$ when $u \sqsubseteq v$ and $v \not\sqsubseteq u$, that is when $u \sqsubseteq v$ and $|u| < |v|$.

When C is a finite index set, $\Sigma^{*C} = \{U, V, \ldots\}$ is the set of mappings from C to Σ^*, i.e. the set of C-indexed tuples of Σ-words. Concatenation and subword ordering extend to tuples from Σ^{*C} in the obvious way.

2.1 (Perfect) channel systems

In this paper we adopt the *extended* model of (lossy) channel systems *where emptiness of channels can be tested for* (and where several messages can be read and written on several channels in a single step). Testing channels for emptiness was allowed in [Sch01] (inspired by [May00]) and we observed that known decidability results do not depend on whether this extension is allowed or not. This remains the case in this paper and the reader will observe that our undecidability proofs do not rely on the extension.

Definition 2.1 (Channel system). *A* channel system *(with n components and m channels) is a tuple $S = \langle \Sigma, C, A_1, A_2, \ldots, A_n \rangle$ where*
- $\Sigma = \{a, b, \ldots\}$ *is a finite alphabet of* messages*,*
- $C = \{c_1, \ldots, c_m\}$ *is a finite set of m* channels*,*
- *for $1 \leq k \leq n$, $A_k = \langle Q_k, \Delta_k \rangle$ is the kth* component *of the system:*
  - $Q_k = \{r, s, \ldots\}$ *is a finite set of* control states*,*
  - $\Delta_k \subseteq Q_k \times \Sigma^{*C} \times Q_k \times \Sigma^{*C} \cup Q_k \times C \times Q_k$ *is a finite set of* rules*.*

A rule $\delta \in \Delta_k$ of the form (s, U, r, V) is written $s \xrightarrow{?U\,!V} r$ and means that A_k can move from s to r by consuming U (i.e. consuming $U(c)$ on each channel $c \in C$) and writing V ($V(c)$ on each c). This assumes that U is available in the channels. A rule of the form (s, c, r) is written $s \xrightarrow{c=\varepsilon?} r$ and means that A_k can move from s to r after checking that channel c is empty.

Formally, the behavior of S is given via a transition system: a *global state* of S is a tuple $\sigma \in Q_1 \times \cdots \times Q_n$ of control states, one for each component of S.

For $1 \le k \le n$, we let $\sigma(k)$ denote the kth component of σ. A *configuration* of S is a pair (σ, W) of a global state and a channel contents $W \in \Sigma^{*C}$ ($W(c) = u$ means that c contains u).

The possible moves between configurations are given by the rules of S. For two configurations (σ, W) and (σ', W') of S we write $\sigma, W \xrightarrow{k:\delta}_{\text{perf}} \sigma', W'$ when:

– δ is some $r \xrightarrow{?U\,!V} s$, $\sigma(k) = r$, there is a W'' s.t. $W = UW''$ and $W' = W''V$. Furthermore $\sigma' = \sigma[k \mapsto s]$, i.e. $\sigma'(k) = s$ and $\sigma'(i) = \sigma(i)$ for all $i \neq k$.

– δ is some $r \xrightarrow{c=\varepsilon?} s$, $\sigma(k) = r$, $W(c) = \varepsilon$ (and further $W' = W$ and $\sigma' = \sigma[k \mapsto s]$).

We write $\sigma, W \xrightarrow{k}_{\text{perf}}$ and say that A_k *is enabled in configuration* (σ, W) when there exists some $\sigma, W \xrightarrow{k:\delta}_{\text{perf}} \sigma', W'$. Otherwise we say A_k is not enabled and write $\sigma, W \not\xrightarrow{k}_{\text{perf}}$.

2.2 Lossy channel systems

The notation "$\rightarrow_{\text{perf}}$" stresses that we just defined **perf**ect steps, i.e. steps where no message is lost. Lossy channel systems are channel systems where steps need not be perfect. Instead, any number of messages can be lost from the channels, without any notification.

In Abdulla and Jonsson's model a lossy step is a perfect step possibly preceded and followed by arbitrary losses from the channels. Formally, we write $\sigma, W \xrightarrow{k:\delta}_{\text{loss}} \sigma', W'$ when there exist channel contents V and V' s.t. $W \sqsupseteq V$, $\sigma, V \xrightarrow{k:\delta}_{\text{perf}} \sigma, V'$ and $V' \sqsupseteq W'$. Perfect steps are lossy steps (with no losses). Below we omit writing explicitely the loss subscript for lossy steps, and are simply careful of writing $\rightarrow_{\text{perf}}$ for all perfect steps.

A *run* π of S (from some initial configuration (σ_0, W_0) often left implicit) is a maximal sequence of steps, of the form $\sigma_0, W_0 \xrightarrow{k_1:\delta_1} \sigma_1, W_1 \xrightarrow{k_2:\delta_2} \sigma_2, W_2 \xrightarrow{k_3:\delta_3} \sigma_3, W_3 \cdots$ Maximality implies that π is either infinite, or finite and ends with a blocked configuration, i.e. a configuration from which no more step is possible. A *perfect* run (also, a *faithful run*) is a run where all steps are perfect (no losses).

By "termination", we mean the absence of any infinite run starting from some given initial configuration. We recall that

Theorem 2.2 ([AJ96b,Fin94]). *Termination is decidable for lossy channel systems.*

2.3 Fair scheduling

There exist many different notions of fairness [Fra86]. Here we consider *fair scheduling of the components*, which is the most natural fairness assumption for asynchronous protocols.

A run of some system S is obtained by interleaving steps from the different components $A_1, \ldots, A_n$. The intuition is that a fair run is a run where all components are fairly treated in their contribution to the run. Formally, given an infinite run $\pi = \sigma_0, W_0 \xrightarrow{k_1:\delta_1} \sigma_1, W_1 \xrightarrow{k_2:\delta_2} \cdots$, we say that:

– π is *weakly fair w.r.t. component* k iff either $k_i = k$ for infinitely many i, or $\sigma_i, W_i \overset{k}{\not\rightarrow}_{\text{perf}}$ for infinitely many i. That is, iff component A_k moves infinitely often in π, or is infinitely often not enabled.

– π is *strongly fair w.r.t. component* k iff either $k_i = k$ for infinitely many i, or $\sigma_i, W_i \overset{k}{\not\rightarrow}_{\text{perf}}$ for almost all i. That is, iff component A_k moves infinitely often in π, or is eventually never enabled.

Additionaly, all finite runs are (vacuously) fair. We say a run is *weakly fair* (resp. *strongly fair*) if it is weakly (resp. strongly) fair w.r.t. all components $A_1, \ldots, A_n$ of S. Clearly, a strongly fair run is also weakly fair.

Remark 2.3. Observe that we do not consider that a component is enabled when it can only perform lossy steps. This definition makes our decidability proofs a bit more involved, but we find it more consistent with the role losses may or may not play in the fairness of scheduling.

2.4 Communication layouts

The *communication layout*, or more simply "the layout", of a channel system $S = \langle \Sigma, C, A_1, \ldots, A_n \rangle$ is a graph depicting which components read from, and write to, which channels. Formally $L(S)$ is the bipartite directed graph having the channels and the components of S as vertices, having an edge from A_k to c if there is a rule $r \xrightarrow{?U\,!V} s$ in Δ_k that writes to c (i.e. $V(c) \neq \varepsilon$), and an edge from c to A_k if there is a rule in Δ_k that reads from c (i.e. $U(c) \neq \varepsilon$). Additionaly, $L(S)$ has an edge from c to A_k if Δ_k has a rule $r \xrightarrow{c=\varepsilon?} s$ that checks c for emptiness.

For example, L_1 in Fig. 2 is the layout of the system from Fig. 1. Note that

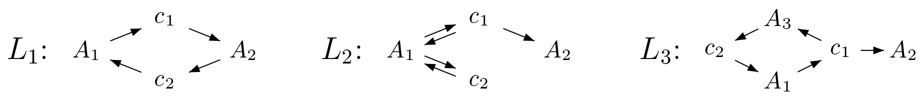

Fig. 2. Three communication layouts

such a layout only describes possible reads and writes (those present in the rules) that are not necessarily actual reads and writes from actual runs.

The layouts of channel systems provide an abstract view of their architecture and are helpful in classifying them. Below we say that a channel c is *multiplexed* if two (or more) components read from it. E.g. any system having L_2 or L_3 (from Fig. 2) as layout has a multiplexed channel since two components read from c_1. (Observe that situations where several components *write* to a same channel are not considered a case of multiplexing.) Many systems have a simple layout like L_1 and have no multiplexed channel. Also many systems use different channels for connecting different sender-receiver pairs, leading to layouts without multiplexed channel.

3 Fair termination without multiplexed channel

For a system S, *strongly fair termination* (resp. weakly) is the property that S has no strongly (resp. weakly) fair infinite run.

Theorem 3.1. *Strongly fair termination and weakly fair termination are decidable for lossy channel systems without multiplexed channel.*

This first positive result is a consequence of the fact that, for systems without multiplexed channel, termination is insensitive to whether the system is fairly scheduled or not (and is therefore decidable by Theorem 2.2). This is proved in Lemma 3.3 after we introduce the necessary definitions.

Definition 3.2. *A lossy channel system S is* insensitive to fairness *(for termination) if the equivalences "S has a strongly fair infinite run iff S has a weakly fair infinite run iff S has an infinite run" hold.*
A communication layout L is insensitive to fairness *if all systems having L as layout are insensitive to fairness.*

Lemma 3.3. *If $S = \langle \Sigma, C, A_1, \ldots, A_n \rangle$ is a system with no multiplexed channel, then S is insensitive to fairness.*

Proof. Assume $\pi = \sigma_0, W_0 \xrightarrow{k_1:\delta_1} \sigma_1, W_1 \xrightarrow{k_2:\delta_2} \sigma_2, W_2 \ldots$ is an infinite run of S. Let $I \subseteq \{1, \ldots, n\}$ be the set of (indexes of) components that are not treated strongly fairly in π, i.e. $k \in I$ iff $k = k_i$ for finitely many i and A_k is infinitely often enabled along π. We let $C_I \subseteq C$ be the set of channels that are read by components in I. Let $l \in \mathbb{N}$ be large enough so that $k_i \notin I$ for all $i \geq l$ and let π' be π where every W_i for $i \geq l$ has been replaced by $W'_i \stackrel{\text{def}}{=} W_i[C_I \mapsto \varepsilon]$, a variant of W_i where channels from C_I have been emptied. π' is a valid run of S since losses can explain the changes in the channel contents, and since only components from I (that never move after l) would have been affected by these changes. We write I' for the set of components that are not treated strongly fairly in π' (observe that $I' \subseteq I$). Now we can build a strongly fair run π'' by inserting, for every $k \in I'$, a step $\sigma_i, W_i \xrightarrow{k:\delta} \sigma'_i, W_i$ at a position i beyond l where A_k is enabled (one such position exists). Such a step does not change W_i (possibly by losing what δ would write) but it modifies $\sigma_i(k)$ and we propagate this change of A_k's control state on all further σ_j. If, when in state $\sigma_i(k)$, A_k is still not treated strongly fairly, we repeat our procedure and insert further steps by A_k. The limit of this construction (that possibly requires an infinite number of insertions) is an infinite strongly fair π''. $\square$

Theorem 3.1 is an important decidability result since systems without multiplexed channel are natural and very common. In fact, these systems are so common (see the examples in [ABJ98,AAB99]) that we feel allowed to claim that, in most practical cases, termination of lossy channel systems does not depend on fair scheduling.

In Section 5 we show more precisely how the absence of multiplexed channels is a necessary condition for Theorem 3.1. Before that, we show the undecidability of strongly fair and weakly fair termination in the general case.

Theorem 3.4. *Strongly fair termination is undecidable for lossy channel systems whose communication layout contains* $A_1 \rightleftarrows c \to A_2$.

Proof. With a Turing machine M we associate the system S_M depicted in Fig. 3. The intended behavior of S_M is the following: first A_1 fills c with some number

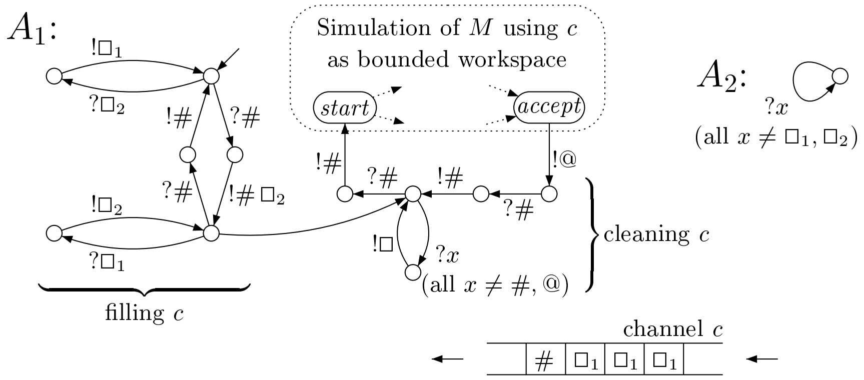

Fig. 3. Structure of S_M in Theorem 3.4

of blank symbols $\Box_1$ using one $\#$ to mark the intended beginning of the string: adding one $\Box_1$ requires two full rotations of the contents of c, replacing all $\Box_1$'s with $\Box_2$'s and then replacing the $\Box_2$'s by $\Box_1$'s. Then A_1 non-deterministically decides that c is full enough and proceeds to the cleaning state (where the $\Box_2$'s are replaced by plain $\Box$'s) that prepares for the *start* state where M is simulated using the contents of c as a bounded workspace, until the *accept* state is eventually reached (if M accepts). At this stage, S_M writes a parasitic character @ on its channel, replaces every other letter by a $\Box$ (i.e. cleans the contents of c) and starts the simulation anew. A_2 does nothing useful but it can consume from c (and will eventually under fair scheduling).

We claim that S_M has a strongly fair run iff M accepts, which proves undecidability. Clearly, if M accepts, S_M has fair infinite runs where it fills c with enough blanks before simulating M an infinite number of times. The parasitic @ that comes up between two successful simulations of M will be either removed by losses, or consumed by A_2 as a way to ensure strong fairness.

Now the more delicate part is to prove that if S_M has a fair infinite run then M accepts. So we assume there is a strongly fair run π (possibly lossy). This run has to eventually move to the *start* state: indeed, if π avoids the *start* state forever, then strong fairness implies that the $\#$ marker will eventually be read by A_2 and then the system will block (thanks to the $\Box_1 \leftrightarrow \Box_2$ swaps). Once π starts simulating M, the contents of c cannot increase in size: it will diminish through losses (or through reads from A_2). After some time, a lower bound is reached and no more loss will ever occur. From now on, strong fairness can only be ensured by having A_2 read the parasitic @, so that the simulation from *start* to *accept* must be performed an infinite number of times. Since no loss occurs, these simulations of M are faithful and prove that M accepts. □

Remark 3.5. The above proof does not describe in more details how the space-bounded Turing machine is simulated because this is standard (since [BZ81]), and because we do not really need to use Turing machines anyway: it is possible to reduce from perfect channel systems. Replace the simulation of M in Fig. 3 by a single-component single-channel system S_1 that works in bounded space (it does not modify the number of messages stored in c). Then the system we built has a strongly fair infinite run iff there exists a number m s.t. S_1, *running as a perfect channel system*, accepts when started with m messages in its channel.

Theorem 3.6. *Weakly fair termination is undecidable for lossy channel systems with two channels.*

Proof (sketch). We prove undecidability for systems whose layout contains the pattern L_2 (from Fig. 2). As in the proof of Theorem 3.4, we associate a system S_M with a Turing machine M in such a way that S_M has a weakly fair infinite run iff M accepts. Here filling c can only proceed as long as d contains one $\#_1$

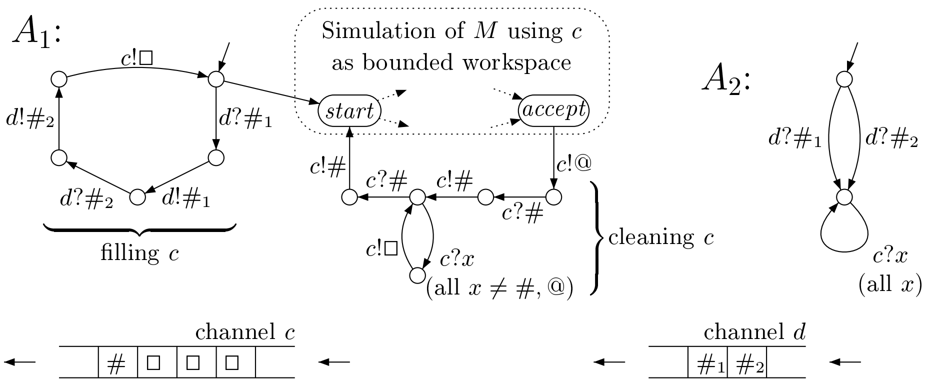

Fig. 4. Structure of S_M in Theorem 3.6

and one $\#_2$, but since A_2 can always read one of these characters, weak fairness requires that, eventually, filling c stops and S_M proceeds to the simulation of M. From this point, the reasoning goes on as in the earlier proof. □

4 Weakly fair termination for single-channel systems

Theorem 4.1. *Weakly fair termination is decidable for systems with one single channel.*

Observe that, by Theorem 3.4, *strongly fair* termination is not decidable in general for single-channel systems, and that, by Theorem 3.6, weakly fair termination is not decidable for systems with *two channels.*

We point out that Theorem 4.1 is not a consequence of insensitivity to fairness. Indeed, there exist single-channel systems having only unfair infinite runs: as an example, consider the system from Fig. 4, restrict A_1 to the loop that fills c and forget about c (only keep d). One obtains a single-channel system that does not terminate unless weakly fair scheduling is assumed.

We now proceed with the proof of Theorem 4.1.

Consider some fixed single-channel system $S = \langle \Sigma, \{c\}, A_1, \ldots, A_n \rangle$ and an initial configuration σ_0, w_0. We say an infinite run $\pi = \sigma_0, w_0 \xrightarrow{k_1:\delta_1} \sigma_1, w_1 \xrightarrow{k_2:\delta_2} \cdots$ is a *bounded run* if there exists some maximal size $K \in \mathbb{N}$ s.t. $|w_i| \leq K$ for all i. Otherwise π is *unbounded.* We say π is *ultimately periodic* if there are two numbers $l, p > 0$ s.t. $\sigma_i, w_i, k_i, \delta_i = \sigma_{i+p}, w_{i+p}, k_{i+p}, \delta_{i+p}$ for all $i \geq l$.

We say a step $\sigma, w \xrightarrow{k:\delta} \sigma', w'$ is *back-lossy* [3] if δ is some $r \xrightarrow{?u\,!v} s$, w is some uu' and w' is $u'v'$ for some $v' \sqsubseteq v$ (that is, losses may only occur during the writing of v at the back of c, and not inside c). Also, all steps with δ of the form $r \xrightarrow{c=\varepsilon?} s$ are (vacuously) back lossy. A run is *back-lossy* if all its steps are. It is *ultimately back-lossy* if after some point all its steps are back-lossy.

The next three lemmas exhibit a sequence of transformations that yield an ultimately periodic weakly fair run out of an unbounded run, entailing Corollary 4.5. The proofs of these lemmas rely on the same extraction and modification techniques on runs we used earlier.

Lemma 4.2. *If S has an unbounded run, then it has an unbounded run that is ultimately back-lossy.*

Proof. Let $\pi = \sigma_0, w_0 \xrightarrow{k_1:\delta_1} \cdots$ be an unbounded run of S.

If π has infinitely many reads, every single letter in every w_i will eventually be lost or consumed. By removing the "to be lost" letters (save those that were already in w_0) we obtain an ultimately back-lossy π'. This does not conclude the proof because π' may be bounded (messages are lost earlier in π' than in π).

If π' is unbounded we are done. Otherwise, this means that there exists a bound $K \in \mathbb{N}$ s.t. every w_i contains at most K letters "to be consumed". Since π is unbounded, for any $M \in \mathbb{N}$ there is some w_l long enough so that it contains at least M consecutive of these "to be lost" letters. If we assume these letters were not already in w_0 (i.e. l is far enough) and that M is larger than $K + 1$ times the number of global states times the length of the longest v written by any one rule, then writing these M consecutive letters in c required that π has a sequence of writes uninterrupted by reads (or emptiness tests) longer than the number of global states. Hence we find a loop of writes that can be repeated to yield an unbounded back-lossy run.

[3] The terminology "back-lossy" appears in [Sch01] and was inspired by the front-lossy systems of [CFP96].

The case where π only reads a finite number of times is simpler (and uses similar ideas). $\square$

Lemma 4.3. *If S has an unbounded run that is ultimately back-lossy, then it has a weakly fair infinite run (perhaps not back-lossy).*

Proof. Assume $\pi = \sigma_0, w_0 \xrightarrow{k_1:\delta_1} \cdots$ is an unbounded back-lossy infinite run and write $I \subseteq \{1, \ldots, n\}$ for the set of components that are not treated weakly fairly.

If π only contains finitely many steps that actually consume message from c, then it is easy to modify π so that we obtain a weakly fair π': a procedure similar to what we did in the proof of Lemma 3.3 is enough, and it can be implemented since, beyond some position l, steps originally in π do not consume from c and are not perturbed when we empty it.

Otherwise π contains infinitely many steps that consume from c and every message in the w_i's will eventually be read, not lost. We call *readee* any $u \in \Sigma^*$ that appears in some rule $r \xrightarrow{?u\,!v} s$ of S. Since π is back-lossy, every w_i can be written under the form $u_{i,1} \ldots u_{i,n_i} v_i$ where the $u_{i,j}$'s are readees (that will eventually be consumed as such along π) and v_i is some suffix that is not yet a full readee. We further decorate w_i by inserting after every $u_{i,j}$ the global state that S will reach just after $u_{i,j}$ is consumed, obtaining some γ_i of the form $u_{i,1}(\rho_{i,1})u_{i,2}(\rho_{i,2}) \ldots u_{i,n_i}(\rho_{i,n_i})v_i$.

We extract from $(\gamma_i)_{i=0,1,\ldots}$ an infinite subsequence along which the size always increases (possible since π is unbounded). Using Higman's lemma, we further extract an infinite subsequence linearly ordered by a variant of the subword ordering where we take as letters the pairs $u(\rho)$ (and the strict prefixes of readees that may occur as v_i's). From this we pick a pair γ_l and $\gamma_{l'}$ s.t. $\gamma_l \sqsubseteq \gamma_{l'}$. We pick l large enough so that components from I are never fired after l, and l' large enough so that all readees in w_l are consumed when moving from σ_l, w_l to $\sigma_{l'}, w_{l'}$ and all components not in I are fired or not enabled at least once between l and l' (and so that $w_{l'}$ is long enough, see below). Here γ_l has the form $u_{l,1}(\rho_{l,1}) \ldots u_{l,n_l}(\rho_{l,n_l})v_l$ and $\gamma_{l'}$ is some $\alpha_0 u_{l,1}(\rho_{l,1})\alpha_1 \ldots \alpha_{l-1} u_{l,n_l}(\rho_{l,n_l})\alpha_l v_l$ where the α_i's witness that $\gamma_l \sqsubseteq \gamma_{l'}$.

If $w_{l'}$ is long enough, then at least one of the α_i's (say α_p) must be longer than the longest readee. Just after the corresponding u_p has been consumed, the steps from σ_l, w_l to $\sigma_{l'}, w_{l'}$ in π visit some intermediary configuration $\sigma_{l''}, w_{l''}$ where $\sigma_{l''} = \rho_{l,p}$ and $w_{l''} = \alpha_p u_{l,p+1}\alpha_{p+1}u_{l,p+1} \ldots$ We now build a looping sequence $\sigma_{l'}, w_{l'} \to \ldots \to \sigma', w_{l'}$ that reuses the rules $\delta_{l+1}, \delta_{l+2}, \ldots, \delta_{l''}$ and that treats all components fairly. This is done by using losses to get rid of the α_i's that appear at the head of c. But when α_p appears, we insert a step by any $A_k \in I$: indeed, π must contain a configuration after l' of the form $(\rho_{l,p}, \alpha_p \ldots)$ and we know A_k is always enabled along π. This inserted step does not consume $u_{l,p+1}$ (since α_p is long enough) and it is then possible to go on with the loop. Note that the control state of A_k has changed to some new s, and all further configurations must be updated, so that σ' is $\sigma_{l'}[k \mapsto s]$. This does not compromise the firability of $\delta_{l+1} \cdots \delta_{l''}$. We can repeat this process and insert steps for any component from

I, and we repeat this infinitely often if needs be. The limit of the construction is a weakly fair infinite run. □

Lemma 4.4. *If S has a weakly fair infinite run, then it has a weakly fair infinite run that is ultimately periodic.*

Proof. This is standard using Higman's lemma: we find two positions l_1 and l_2 s.t. $\sigma_{l_1} = \sigma_{l_2}$ and $w_{l_1} \sqsubseteq w_{l_2}$. We further pick l_1 and l_2 large enough so that all components are fired or not enabled at least once between l_1 and l_2. Losses after step l_2 allow to reach σ_{l_2}, w_{l_1}, closing the loop for an ultimately periodic run and maintaining weak fairness. □

Corollary 4.5. *Either S only has bounded runs, or it has an ultimately periodic weakly fair infinite run (or both).*

Now the proof of Theorem 4.1 is easy: Boundedness of single-channel systems is not decidable, but it is obviously semi-decidable, and for a bounded S weakly fair termination is easily checked after the finite graph of configurations has been constructed. Similarly, the existence of a weakly fair ultimately periodic run π is easily seen to be semi-decidable since it suffices to exhibit a finite prefix of π. Combining these two semi-decision methods, we obtain a decision algorithm.

5 Classifying communication layouts

In this section, we characterize the layouts that induce decidability of strongly fair termination.

Let L be a communication layout. We say that L *has multiplexing inside a cycle* iff there exists a multiplexed channel c that lies on a (directed) cycle in L.

Theorem 5.1. *Strongly fair termination of systems having communication layout L is decidable iff L does not have multiplexing inside a cycle.*

Proving Theorem 5.1 requires that we complement the results from section 3 with the following key decomposition lemma.

Let L be a layout s.t. some channel c does not lie on a (directed) cycle in L. Then L can be seen as $L_1 \oplus_c L_2$, i.e. the gluing via c of two disjoint layouts L_1 and L_2 (both of them containing c), as illustrated in Fig. 5.

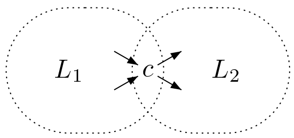

Fig. 5. L is $L_1 \oplus_c L_2$

Lemma 5.2. *L is insensitive to fairness iff L_1 and L_2 are.*

Proof. We only need to prove the ($\Leftarrow$) direction. For this we consider a system S with $L(S) = L$, and prove S is insensitive to fairness. Let S_1 and S_2 be the subsystems obtained from S by keeping only the components (and the channels) from L_1 (resp. L_2). Let π be an infinite run by S. There are two cases:

π contains infinitely many steps from S_1: then S_1 has an infinite run (since steps by S_2 cannot influence S_1) and it has an infinite fair run (since it is insensitive to fairness). Inserting as many steps by S_2 as necessary, one turns this run into a π' that is fair w.r.t. all components of S.

π contains finitely many steps from S_1: then π can be written as the concatenation $\pi_1.\pi_2$ of a finite prefix π_1 where all steps by S_1 can be found, followed by an infinite run π_2 of S_2 (from some starting configuration). By insensitivity, π_2 can be replaced by a fair π_2' (fair w.r.t. S_2). We obtain a run fair w.r.t. all of S by inserting in π_2' (that is, after π_1) as many steps by S_1 as necessary, using losses to make sure these extra steps do not add to c. $\square$

Corollary 5.3. *Layouts without multiplexing inside a cycle are insensitive to fairness.*

Proof. By induction on the number of multiplexed channels in the layout. Lemma 5.2 lets us reduce to the base case where the multiplexed channels (if any) are *degenerate*, i.e. no component writes to them (e.g. in the above picture, c is degenerate in L_2). Insensitivity for systems with degenerate multiplexed channels is proved exactly like with Theorem 3.1. $\square$

Proof (of Theorem 5.1). The ($\Leftarrow$) direction was proved as Corollary 5.3. The ($\Rightarrow$) direction is an easy extension of Theorem 3.4. Assume that L has a cycle $A_1 \to c_1 \to A_2 \to c_2 \cdots A_n \to c_n \to A_1$ s.t. one channel, say c_n, is multiplexed. Then c_n is read by some component A distinct from A_1. If A itself is not on the cycle, then it is easy to adapt the proof of Theorem 3.4 and prove undecidability. Otherwise A is some A_i for $i > 1$ and we can find a shorter cycle $A_i \to c_i \to A_{i+1} \cdots A_n \to c_n \to A_i$, where this time the outside component A is A_1, and we conclude as before. $\square$

Remark 5.4. Lemma 5.2 and Corollary 5.3 apply to strong and weak fairness equally. The reason why the characterization provided by Theorem 5.1 does not hold for weakly fair termination is that Theorem 3.4 only deals with strong termination (which cannot be avoided, see Theorem 4.1).

6 Other verification problems with fair scheduling

Termination is not the only verification problem that is known to be decidable for lossy channel systems, but problems like reachability only consider *finite* runs. The other known decidable problem for which fairness assumptions are meaningful is *inevitability* (shown decidable in [AJ96b]). Here one asks whether all runs eventually visit a configuration belonging to a given set G.

Termination is a special case of inevitability (with G being the set of blocked configurations), but our positive results for fair termination do not generalize to fair inevitability:

Theorem 6.1. *Inevitability under strongly fair or weakly fair scheduling is undecidable for systems with (more than one component and) a communication layout containing $A_1 \rightleftarrows c$.*

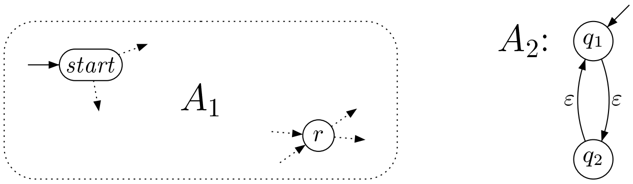

Fig. 6. Reducing RCS to fair inevitability

Proof (sketch). By a reduction from RCS. Let A_1 be any given component having some control state r. We build a system S by associating A_1 and the system A_2 from Fig. 6 (observe that A_2 does not use the channel and is always enabled). Now let G be the set of all configurations $(\langle s, s' \rangle, w)$ of S s.t. $s \neq r$ and $s' = q_2$. Then all fair runs of S inevitably visit G iff A_1 does not have a run visiting r infinitely often (unless there is a self-loop on r). ☐

This proof idea can be adapted to layouts that contains a cycle, so that inevitability under fair scheduling is undecidable for all "interesting" layouts.

7 Conclusions

We studied the decidability of termination under strongly and weakly fair scheduling. We showed that, when systems have no multiplexed channels, termination does not depend on whether scheduling is fair or not. In practice, most systems do not have multiplexed channels since they use distinct channels for any pair of components that communicate.

We also showed that, for systems where an arbitrary number of components communicate through a single channel, weakly fair termination is decidable.

These results are technically involved, and are close to the border of decidability. Indeed, two channels make weakly fair termination undecidable, and strongly fair termination is decidable iff no multiplexed channel occurs inside a communication cycle.

References

[AAB99] P. A. Abdulla, A. Annichini, and A. Bouajjani. Symbolic verification of lossy channel systems: Application to the bounded retransmission protocol. In *Proc. 5th Int. Conf. Tools and Algorithms for the Construction and Analysis of Systems (TACAS'99), Amsterdam, The Netherlands, Mar. 1999*, volume 1579 of *Lecture Notes in Computer Science*, pages 208–222. Springer, 1999.

[ABJ98] P. A. Abdulla, A. Bouajjani, and B. Jonsson. On-the-fly analysis of systems with unbounded, lossy FIFO channels. In *Proc. 10th Int. Conf. Computer Aided Verification (CAV'98), Vancouver, BC, Canada, June-July 1998*, volume 1427 of *Lecture Notes in Computer Science*, pages 305–318. Springer, 1998.

[AJ96a] P. A. Abdulla and B. Jonsson. Undecidable verification problems for programs with unreliable channels. *Information and Computation*, 130(1):71–90, 1996.

[AJ96b] P. A. Abdulla and B. Jonsson. Verifying programs with unreliable channels. *Information and Computation*, 127(2):91–101, 1996.

[AK95] P. A. Abdulla and M. Kindahl. Decidability of simulation and bisimulation between lossy channel systems and finite state systems. In *Proc. 6th Int. Conf. Theory of Concurrency (CONCUR'95), Philadelphia, PA, USA, Aug. 1995*, volume 962 of *Lecture Notes in Computer Science*, pages 333–347. Springer, 1995.

[BZ81] D. Brand and P. Zafiropulo. On communicating finite-state machines. Research Report RZ 1053, IBM Zurich Research Lab., June 1981. A short version appears in J.ACM 30(2):323–342, 1983.

[CFP96] G. Cécé, A. Finkel, and S. Purushothaman Iyer. Unreliable channels are easier to verify than perfect channels. *Information and Computation*, 124(1):20–31, 1996.

[Fin94] A. Finkel. Decidability of the termination problem for completely specificied protocols. *Distributed Computing*, 7(3):129–135, 1994.

[Fra86] N. Francez. *Fairness*. Springer, 1986.

[May00] R. Mayr. Undecidable problems in unreliable computations. In *Proc. 4th Latin American Symposium on Theoretical Informatics (LATIN'2000), Punta del Este, Uruguay, Apr. 2000*, volume 1776 of *Lecture Notes in Computer Science*, pages 377–386. Springer, 2000.

[Sch01] Ph. Schnoebelen. Bisimulation and other undecidable equivalences for lossy channel systems. In *Proc. 4th Int. Symp. Theoretical Aspects of Computer Software (TACS'2001), Sendai, Japan, Oct. 2001*, volume 2215 of *Lecture Notes in Computer Science*, pages 385–399. Springer, 2001.

[Sch02] Ph. Schnoebelen. Verifying lossy channel systems has nonprimitive recursive complexity. *Information Processing Letters*, 83(5):251–261, 2002.