Mixing lossy and perfect fifo channels*
(Extended abstract)

P. Chambart and Ph. Schnoebelen

LSV, ENS Cachan, CNRS
61, av. Pdt. Wilson, F-94230 Cachan, France
email: {chambart|phs}@lsv.ens-cachan.fr

Abstract. We consider asynchronous networks of finite-state systems communicating via a combination of reliable and lossy fifo channels. Depending on the topology, the reachability problem for such networks may be decidable. We provide a complete classification of network topologies according to whether they lead to a decidable reachability problem. Furthermore, this classification can be decided in polynomial-time.

1 Introduction

Fifo channels. Channel systems, aka "communicating finite-state machines", are a classical model for protocols where components communicate asynchronously via fifo channels [8]. When the fifo channels are unbounded, the model is Turing-powerful since channels can easily be used to simulate the tape of a Turing machine.

It came as quite a surprise when Abdulla and Jonsson [4, 3], and independently Finkel *et al.* [13], showed that *lossy* channel systems (LCS's), i.e., channel systems where one assumes that the channels are unreliable so that messages can be lost nondeterministically, are amenable to algorithmic verification (see also [20]). The model has since been extended in several directions: message losses obeying probability laws [21, 1, 2, 6], channels with other kinds of unreliability [9, 7], etc.

How this unreliability leads to decidability is paradoxical, and hard to explain in high-level, non-technical terms. It certainly does not make the model trivial: we recently proved that LCS verification is exactly at level $\mathcal{F}_{\omega^\omega}$ in the Extended Grzegorczyk Hierarchy, hence it is not primitive-recursive, or even multiply-recursive [12].

An ubiquitous model. In recent years, lossy channels have shown up in unexpected places. They have been used in reductions showing hardness (or less frequently decidability) for apparently unrelated problems in modal logics [16], in temporal logics [19], in timed automata [17], in data-extended models [15], etc. More and more, LCS's appear to be a pivotal model whose range goes far beyond asynchronous protocols.

Fueling this line of investigation, we recently discovered that the "Regular Post Embedding Problem", a new decidable variant of Post's Correspondence Problem, is equivalent (in a non-trivial way) to LCS reachability [10, 11]. This discovery was an

* Work supported by the Agence Nationale de la Recherche, grant ANR-06-SETIN-001.

unexpected outcome of our study of *unidirectional* channel systems (UCS), where a Sender can send messages to a Receiver via two fifo channels, one reliable and one lossy, but where there is no communication in the other direction (see T_2^d in Fig. 1 below). As far as we know, this simple arrangement had never been studied before.

Our contribution. This paper considers the general case of *mixed* channel systems, where some channels are reliable and some are lossy. These systems can be Turing-powerful (one process using one reliable fifo buffer is enough) but not all network topologies allow this (e.g., systems with only lossy channels, or systems where communication is arranged in a tree pattern with no feedback, or UCS's as above). We provide a complete classification of network topologies according to whether they lead to undecidable reachability problems, or not. This relies on original and non-trivial transformation techniques for reducing large topologies to smaller ones while preserving decidability.

Beyond providing a complete classification, the present contribution has several interesting outcomes. First, we discovered new decidable arrangements of channel systems, as well as new undecidable ones, and these new results are often surprising. They enlarge the existing toolkit currently used when transferring results from channel systems to other areas, according to the "ubiquitous model" slogan. Secondly, the transformation techniques we develop may eventually prove useful for reducing/delaying the combinatorial explosion one faces when verifying asynchronous protocols.

Outline of the paper. We describe *mixed channel systems* and their topologies in Section 2 and provide in Section 3 a few original results classifying the basic topologies to which we reduce larger networks. Section 4 shows that "fusing essential channels" preserves decidability. An additional "splitting" technique is described in Section 5. After these three sections, we have enough technical tools at hand to describe our main result, the complete classification method, and prove its correctness in Sections 6 and 7.

2 Systems with reliable and lossy channels

We classify channel systems according to their *network topology*, which is a graph describing who are the participant processes and what channels they are connected to.

2.1 Network topologies

Formally, a *network topology*, or shortly a *topology*, is a tuple $T = \langle N, R, L, s, d \rangle$ where N, R and L are three mutually disjoint finite sets of, respectively, *nodes*, *reliable channels*, and *lossy channels*, and where, writing $C \stackrel{\text{def}}{=} R \cup L$ for the set of channels, $s, d : C \to N$ are two mappings that associate a *source* and a *destination* node to each channel. We do not distinguish between isomorphic topologies since N, R and L simply contain "names" for nodes and channels: these are irrelevant here and only the directed graph structure with two types of edges matters.

Graphical examples of simple topologies will be found below: we use dashed arrows to single out the lossy channels (reliable channels are depicted with full arrows).

2.2 Mixed channel systems and their operational semantics

Assume $T = \langle N, R, L, s, d\rangle$ is a topology with n nodes, i.e., with $N = \{P_1, P_2, ..., P_n\}$. Write $C = R \cup L$ for the set of channels. A *mixed channel system* (MCS) having topology T is a tuple $S = \langle T, M, Q_1, \Delta_1, ..., Q_n, \Delta_n\rangle$ where $M = \{\mathsf{a}, \mathsf{b}, ...\}$ is a finite *message alphabet* and where, for $i = 1, ..., n$, Q_i is the finite set of (control) states of a process (also denoted P_i) that will be located at node $P_i \in N$, and Δ_i is the finite set of *transition rules*, or shortly "rules", governing the behaviour of P_i. A rule $\delta \in \Delta_i$ is either a *writing rule* of the form $(q, c, !, \mathsf{a}, q')$, usually denoted "$q \xrightarrow{c!\mathsf{a}} q'$", with $q, q' \in Q_i$, $s(c) = P_i$ and $\mathsf{a} \in M$, or it is a *reading rule* $(q, c, ?, \mathsf{a}, q')$, usually denoted "$q \xrightarrow{c?\mathsf{a}} q'$", with this time $d(c) = P_i$. Hence the way a topology T is respected by a channel system is via restrictions upon the set of channels to which a given participant may read from, or write to.

Our terminology "*mixed* channel system" is meant to emphasize the fact that we allow systems where lossy channels coexist with reliable channels.

The behaviour of some $S = \langle T, M, Q_1, \Delta_1, ..., Q_n, \Delta_n\rangle$ is given under the form of a transition system. Assume $C = \{c_1, ..., c_k\}$ contains k channels. A configuration of S is a tuple $\sigma = \langle q_1, ..., q_n, u_i, ..., u_k\rangle$ where, for $i = 1, ..., n$, $q_i \in Q_i$ is the current state of P_i, and where, for $i = 1, ..., k$, $u_i \in M^*$ is the current contents of channel c_i.

Assume $\sigma = \langle q_1, ..., q_n, u_i, ..., u_k\rangle$ and $\sigma' = \langle q'_1, ..., q'_n, u'_i, ..., u'_k\rangle$ are two configurations of some system S as above, and $\delta \in \Delta_i$ is a rule of participant P_i. Then δ witnesses a transition between σ and σ', also called a *step*, and denoted $\sigma \xrightarrow{\delta} \sigma'$, if and only if

- the control states agree with, and are modified according to δ, i.e., $q_i = q$, $q'_i = q'$, $q_j = q'_j$ for all $j \neq i$;
- the channel contents agree with, and are modified according to δ, i.e., either
  - $\delta = (q, c_l, ?, \mathsf{a}, q')$ is a reading rule, and $u_l = \mathsf{a}.u'_l$, or
  - $\delta = (q, c_l, !, \mathsf{a}, q')$ is a writing rule, and $u'_l = u_l.\mathsf{a}$, or $c_l \in L$ is a lossy channel and $u'_l = u_l$;

  in both cases, the other channels are untouched: $u'_j = u_j$ for all $j \neq l$.

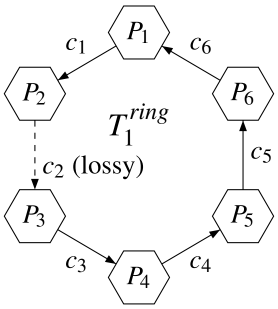
Such a step is called "*a step by P_i*" and we say that its *effect* is "reading a on c", or "writing a to c", or "losing a". A *run* (from σ_0 to σ_p) is a sequence of steps of the form $r = \sigma_0 \xrightarrow{\delta_1} \sigma_1 \xrightarrow{\delta_2} \sigma_2 \cdots \xrightarrow{\delta_p} \sigma_p$, sometimes shortly written $\sigma_0 \xrightarrow{*} \sigma_p$. A run is *perfect* if none of its steps loses a message.

Remark 2.1. With this operational semantics for lossy channels, messages can only be lost when a rule writes them to a channel. Once inside the channels, messages can only be removed by reading rules. This definition is called the *write-lossy* semantics for lossy channels: it differs from the more classical definition where messages in lossy channels can be lost at any time. We use it because it is the most convenient one for our current concerns, and because this choice does not impact the reachability questions we consider (see [12, Appendix A] for a formal comparison). $\square$

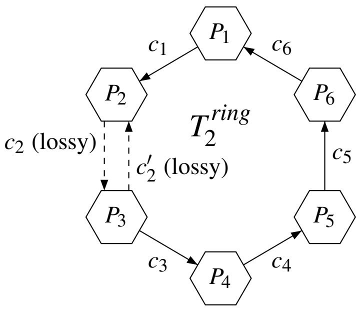
2.3 The reachability problem for network topologies

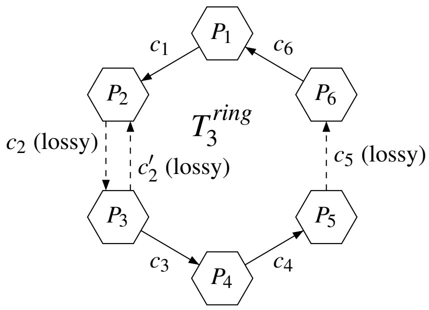
The *reachability problem* for mixed channel systems asks, for a given S and two configurations $\sigma_{\text{init}} = \langle q_1, \ldots, q_n, \varepsilon, \ldots, \varepsilon \rangle$ and $\sigma_{\text{final}} = \langle q'_1, \ldots, q'_n, \varepsilon, \ldots, \varepsilon \rangle$ in which the channels are empty, whether S has a run from σ_{init} to σ_{final}. That we restrict reachability questions to configurations with empty channels (ε denotes the empty word in M^*) is technically convenient, but it is no real loss of generality.

The *reachability problem* for a topology T is the restriction of the reachability problem to mixed systems having topology T. Hence if reachability is decidable for T, it is decidable for all MCS's having topology T. If reachability is not decidable for T, it may be decidable or not for MCS's having topology T (but it must be undecidable for one of them). Clearly, if T' is a subgraph of T and reachability is decidable for T, then it is for T' too.



Our goal is to determine for which topologies reachability is decidable. Let us illustrate the question and outline some of our results. T_1^{ring} is a topology describing a directed ring of processes, where each participant sends to its right-hand neighbour, and receives from its left-hand neighbour. A folk claim is that such cyclic networks have decidable reachability as soon as one channel is lossy (as here with c_2). The proof ideas behind this claim have not been formally published and they do not easily adapt to related questions like "what about T_2^{ring}?", where a lossy channel in the other direction is added, or about T_3^{ring} where more channels are lossy in the ring.





Our techniques answer all three questions uniformly. One of our results states that all channels along the path c_3 to c_4 to c_5 to c_6 to c_1 can be fused into a single channel going from P_3 to P_2 without affecting the decidability of reachability. The transformations are modular (we fuse one channel at a time). Depending on the starting topology, we end up with different two-node topologies, from which we deduce that T_1^{ring} and T_3^{ring} have decidable reachability, while T_2^{ring} does not (see Corollary 4.6 below).

3 Reachability for basic topologies

This section is concerned with the basic topologies to which we will later reduce all larger cases.

Theorem 3.1 (Basic topologies). *Reachability is decidable for the network topologies T_1^d and T_2^d (see Fig. 1). It is not decidable for the topologies T_1^u, T_2^u, T_3^u, T_4^u, T_5^u, and T_6^u (see Fig. 2).*

We start with the decidable cases:

That T_1^d, and more generally all topologies with only lossy channels (aka LCS's), leads to decidable problems is the classic result from [4].

Regarding T_2^d, we recently proved it has decidable reachability in [10], where T_2^d-systems are called "unidirectional channel systems", or UCS's. Our reason for investigating UCS's was indeed that this appeared as a necessary preparation for the classification of mixed topologies. Showing that T_2^d has decidable reachability is quite involved, going through the introduction of the "Regular Post Embedding Problem". In addition, [10, 11] exhibit non-trivial reductions between reachability for UCS's and reachability for LCS's: the two problems are equivalent.

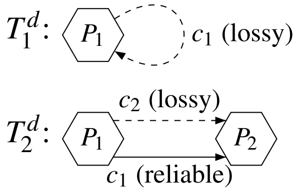


Fig. 1. Basic decidable topologies

Now to the undecidable cases:

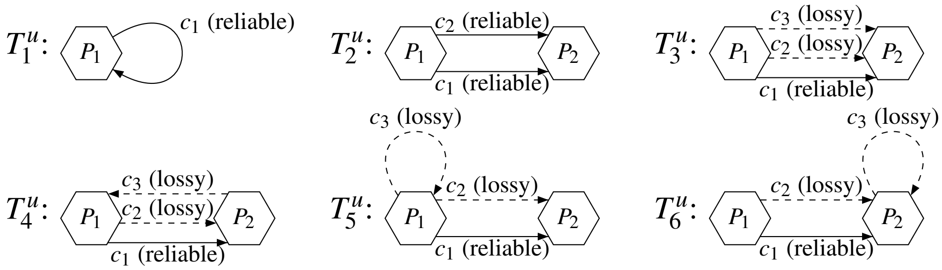


Fig. 2. Basic topologies with undecidable reachability

It is well-known that T_1^u may lead to undecidable problems [8], and this is also known, though less well, for T_2^u (restated, e.g., as the non-emptiness problem for the intersection of two rational transductions). The other four results mix lossy and reliable channels and are new. We actually prove all six cases in a uniform framework, by reduction from Post's Correspondence Problem, aka PCP, or its directed variant, $\mathsf{PCP}_{\mathrm{dir}}$.

Recall that an instance of PCP is a family $x_1, y_1, x_2, y_2, \ldots, x_n, y_n$ of $2n$ words over some alphabet. The question is whether there is a non-empty sequence (a *solution*) $i_1, \ldots, i_k$ of indexes such that $x_{i_1} x_{i_2} \ldots x_{i_k} = y_{i_1} y_{i_2} \ldots y_{i_k}$. $\mathsf{PCP}_{\mathrm{dir}}$ asks whether there is a *directed* solution $i_1, \ldots, i_k$, i.e., a solution such that, in addition, $y_{i_1} y_{i_2} \ldots y_{i_h}$ is a prefix of $x_{i_1} x_{i_2} \ldots x_{i_h}$ for all $h = 1, \ldots, k$. It is well-known that PCP and $\mathsf{PCP}_{\mathrm{dir}}$ are undecidable, and more precisely Σ_0^1-complete.

Reducing PCP *to T_2^u-networks.* With a PCP instance $(x_i, y_i)_{i=1,\ldots,n}$, we associate a process P_1 having a single state p_1 and n loops[1] $p_1 \xrightarrow{c_1!x_i\ c_2!y_i} p_1$, one for each index $i = 1,\ldots,n$. Process P_1 guesses a solution $i_1 i_2 i_3 \ldots$ and sends the concatenations $x_{i_1} x_{i_2} x_{i_3} \ldots$ and $y_{i_1} y_{i_2} y_{i_3} \ldots$ on, respectively, c_1 and c_2. Process P_2 checks that the two channels c_1 and c_2 have the same contents, using reading loops $p_2 \xrightarrow{c_1?\mathsf{a}\ c_2?\mathsf{a}} p_2$, one for each symbol $\mathsf{a}, \mathsf{b}, \ldots$ in the alphabet. An extra control state, for example p_1' with rules $p_1' \xrightarrow{c_1!x_i\ c_2!y_i} p_1$, is required to check that P_1 picks a non-empty solution. Then, in the resulting T_2^u-network, $\langle p_1', p_2, \varepsilon, \varepsilon \rangle \xrightarrow{*} \langle p_1, p_2, \varepsilon, \varepsilon \rangle$ if and only if the PCP instance has a solution.

Reducing PCP *to T_3^u-networks.* For T_3^u, the same idea is adapted to a situation with three channels, two of which are lossy. Here P_1 has rules $p_1 \xrightarrow{c_2!x_i\ c_3!y_i\ c_1!1^{|x_i y_i|}} p_1$. Thus P_1 sends x_i and y_i on lossy channels and simultaneously sends the number of letters in unary (1 is a special tally symbol) on c_1, the perfect channel. P_2 matches these with reading loops of the form $p_2 \xrightarrow{c_1?11\ c_2?\mathsf{a}\ c_3?\mathsf{a}} p_2$ for each letter a. If P_2 can consume all 1's out of c_1, this means that no message has been lost on the lossy channels, and then P_2 really witnessed a solution the PCP instance.

Reducing PCP_{dir} *to T_1^u-networks.* For T_1^u, we consider the directed $\mathsf{PCP}_{\mathrm{dir}}$. P_1 has n loops $p_1 \xrightarrow{c_1!x_i\ c_1?y_i} p_1$ where the guessing and the matching is done by a single process. Since at any step $h = 1, \ldots, k$ the concatenation $x_{i_1} x_{i_2} \ldots x_{i_h}$ is (partly) consumed while matching for $y_{i_1} y_{i_2} \ldots y_{i_h}$, only directed solutions will be accepted.

Reducing PCP_{dir} *to T_5^u-networks.* For T_5^u too, we start from $\mathsf{PCP}_{\mathrm{dir}}$ and use a variant of the previous counting mechanism to detect whether some messages have been lost. P_1 has rules of the form $p_1 \xrightarrow{c_3!1^{|x_i|}\ c_1!x_i\ c_3?1^{|y_i|}\ c_2!y_i} p_1$, i.e., it sends x_i on c_1 (the reliable channel) and y_i on c_2 (unreliable) while P_2 checks the match with loops $p_2 \xrightarrow{c_1?\mathsf{a}\ c_2?\mathsf{a}} p_2$. In addition, P_1 also maintains in c_3 a count of the number of symbols written to c_1 minus the number of symbols written to c_2, or $\#_h \stackrel{\text{def}}{=} |x_{i_1} \ldots x_{i_h}| - |y_{i_1} \ldots y_{i_h}|$. The counting scheme forbids partial sequences $y_{i_1} \ldots y_{i_h}$ that would be longer than the corresponding $x_{i_1} \ldots x_{i_h}$, but this is right since we look for directed solutions. If tally symbols on c_3 are lost, or if part of the y_i's on c_2 are lost, then it will never be possible for P_2 to consume all messages from c_1. Finally a run from $\langle p_1', p_2, \varepsilon, \varepsilon, \varepsilon \rangle$ to $\langle p_1, p_2, \varepsilon, \varepsilon, \varepsilon \rangle$ must be perfect and witness a directed solution.

Reducing PCP_{dir} *to T_6^u-networks.* For T_6^u, we adapt the same idea, this time having P_2 monitoring the count $\#_h$ on c_3. P_1 has loops $p_1 \xrightarrow{c_1!x_i 1^{|y_i|}\ c_2!y_i 1^{|x_i|}} p_1$ where a guessed

[1] Transition rules like "$p_1 \xrightarrow{c_1!x_i\ c_2!y_i} p_1$" above, where several reads and writes are combined in a same rule, and where one writes or reads words rather than just one message at a time, are standard short-hand notations for sequences of rules using intermediary states that are left implicit. We avoid using this notation in situations where the specific ordering of the combined actions is important as, e.g., in $(*)$ below.

solution is sent on c_1 and c_2 with interspersed tally symbols. The guessed solution is checked with the usual loops $p_2 \xrightarrow{c_1?\mathtt{a}\ c_2?\mathtt{a}} p_2$. The 1's on c_2 are stored to c_3 and matched (later) with the 1's on c_1 via two loops: $p_2 \xrightarrow{c_2?1\ c_3!1} p_2$ and $p_2 \xrightarrow{c_3?1\ c_1?1} p_2$. In a perfect run, there are always as many messages on c_1 as there are on c_2 and c_3 together, and strictly more if a message is lost. Hence a run from $\langle p_1', p_2, \varepsilon, \varepsilon, \varepsilon\rangle$ to $\langle p_1, p_2, \varepsilon, \varepsilon, \varepsilon\rangle$ must be perfect and witness a solution. Only direct solutions can be accepted since the tally symbols in c_3 count $\#_h$ that cannot be negative.

Reducing PCP_{dir} *to* T_4^u*-networks.* For T_4^u, we further adapt the idea, again with the count $\#_h$ stored on c_3 but now sent from P_2 to P_1. The loops in P_1 now are

$$p_1 \xrightarrow{c_1!x_i\ c_2!y_i 1^{|x_i|}} q_i \xrightarrow{c_3?1^{|y_i|}} p_1. \tag{$*$}$$

The 1's on c_2 are sent back via c_3 to be matched later by P_1, thanks to a loop $p_2 \xrightarrow{c_2?1\ c_3!1} p_2$. Again a message loss will leave strictly more messages in c_1 than in c_2 and c_3 together, and cannot be recovered from. Only direct solutions can be accepted since the tally symbols in c_3 count $\#_h$.

4 Fusion for essential channels

Sections 4 and 5 develop techniques for "simplifying" topologies while preserving the decidability status of reachability problems. We start with a reduction called "fusion".

Let $T = \langle N, R, L, s, d\rangle$ be a network topology. For any channel $c \in C$, $T - c$ denotes the topology obtained from T by deleting c. For any two distinct nodes $P_1, P_2 \in N$, $T[P_1 = P_2]$ denotes the topology obtained from T by merging P_1 and P_2 in the obvious way: channel extremities are redirected accordingly.

Clearly, any MCS with topology $T - c$ can be seen as having topology T. Thus $T - c$ has decidable reachability when T has, but the converse is not true in general.

Similarly, any MCS having topology T can be transformed into an equivalent MCS having topology $T[P_1 = P_2]$ (using the asynchronous product of two control automata). Thus T has decidable reachability when $T[P_1 = P_2]$ has, but the converse is not true in general.

For any channel c such that $s(c) \neq d(c)$, we let T/c denote $T[s(c) = d(c)] - c$ and say that T/c is "*obtained from T by contracting c*". Hence T/c is obtained by merging c's source and destination, and then removing c.

Since T/c is obtained via a combination of merging and channel removal, there is, in general, no connection between the decidability of reachability for T and for T/c. However, there is a strong connection for so-called "essential" channels, as stated in Theorem 4.5 below.

Before we can get to that point, we need to explain what are essential channels and how they can be used.

4.1 Essential channels are existentially 1-bounded

In this section, we assume a given MCS $S = \langle T, M, Q_1, \Delta_1, \ldots\rangle$ with $T = \langle N, R, L, s, d\rangle$.

Definition 4.1. *A channel $c \in C$ is* essential *if $s(c) \neq d(c)$ and all directed paths from $s(c)$ to $d(c)$ in T go through c.*

In other words, removing c modifies the connectivity of the directed graph underlying T.

The crucial feature of an essential channel c is that causality between the actions of $s(c)$ and the actions of $d(c)$ is constrained. As a consequence, it is always possible to reorder the actions in a run so that reading from c occurs immediately after the corresponding writing to c. As a consequence, bounding the number of messages that can be stored in c does not really restrict the system behaviour.

Formally, for $b \in \mathbb{N}$, we say a channel c is b-bounded along a run $\pi = \sigma_0 \xrightarrow{\delta_1} \ldots \xrightarrow{\delta_n} \sigma_n$ if $|\sigma_i(c)| \leq b$ for $i = 0, \ldots, n$. We say c is *synchronous* in π if it is 1-bounded and at least one of $\sigma_i(c)$ and $\sigma_{i+1}(c)$ is ε for all $0 \leq i < n$. Hence a synchronous channel only stores at most one message at a time, and the message is read immediately after it has been written to c.

Proposition 4.2. *If c is essential and $\pi = \sigma_0 \xrightarrow{\delta_1} \ldots \xrightarrow{\delta_n} \sigma_n$ is a run with $\sigma_0(c) = \sigma_n(c) = \varepsilon$, then S has a run π' from σ_0 to σ_n in which c is synchronous.*

This notion is similar to the existentially-bounded systems of [18] but is applies to a single channel, not to the whole system.

We prove Proposition 4.2 using techniques and concepts from true concurrency theory and message flow graphs (see, e.g., [14]). With a run $\pi = \sigma_0 \xrightarrow{\delta_1} \ldots \xrightarrow{\delta_n} \sigma_n$ as above, we associate a set $E = \{1, \ldots, n\}$ of n events, that can be thought of the actions performed by the n steps of π: firing a transition and reading or writing or losing a message. Observe that different occurrences of a same transition with same effect are two different events. We simply identify the events with indexes from 1 to n. We write $e, e', \ldots$ to denote events, and also use the letters r and w for reading and writing events.

Any $e \in E$ is an event of some process $N(e) \in N$ and we write $E = \bigcup_{P \in N} E_P$ the corresponding partition. There exist several (standard) causality relations between events. For every process $P \in N$, the events of P are linearly ordered by $<_P$: $i <_P j$ iff $i, j \in E_P$ and $i < j$. For every channel $c \in C$, the events that write to or read from c are related by $<_c$ with $i <_c j$ iff i is an event that writes some m to c, and j is the event that reads that (occurrence of) m. (Here, events that lose messages are considered as internal actions where no channel is involved.) We let $\prec$ (and $\preccurlyeq$) denote the transitive (resp. reflexive-transitive) closure of $\bigcup_{P \in N} <_P \cup \bigcup_{c \in C} <_c$. $(E, \preccurlyeq)$ is then a poset, and $\preccurlyeq$ is called the *visual* order (also causality order, or dependency order) in the literature. For $e \in E$, we let $\downarrow e$ denote the past of e, i.e., the set $\{e' \in E \mid e' \preccurlyeq e\}$.

It is well-known that any linear extension $e_1, \ldots, e_n$ of $(E, \preccurlyeq)$ is causally consistent and can be transformed into a run $\pi' = \sigma_0 \xrightarrow{e_1} \xrightarrow{e_2} \cdots$ starting from σ_0. This run ends in σ_n like π, though it may go through different intermediary configurations. All the runs obtained by considering different linear extensions are causally equivalent to π, denoted

$\pi \approx \pi'$, and they all give rise to the same poset $(E, \preccurlyeq)$.

We now state properties enjoyed by $(E, \preccurlyeq)$ in our context that are useful for proving Proposition 4.2. First, observe that, since the channels are fifo, and since only one process, namely $d(c)$ (resp. $s(c)$), is allowed to read from (resp. write to) a channel c:

$$(w_1 <_c r_1 \text{ and } w_2 <_c r_2) \text{ imply } (w_1 <_{s(c)} w_2 \text{ iff } r_1 <_{d(c)} r_2). \tag{$\dagger$}$$

($\dagger$) is sometimes taken as a definition of fifo communication.

Another important observation is the following: assume $e \preccurlyeq e'$. Then, and since $\preccurlyeq$ is defined as a reflexive-transitive closure, there must be a chain of the form

$$\theta: \quad e = e_0 \leq_{P_0} e'_0 <_{c_1} e_1 \leq_{P_1} e'_1 <_{c_2} \ldots <_{c_l} e_l \leq_{P_l} e'_l = e'$$

where, for $1 \leq i \leq l$, $s(c_i) = P_{i-1}$ and $d(c_i) = P_i$. Hence T has a path $c_1, \ldots, c_l$ going from P_0 to P_l.

Lemma 4.3. *If $e_1 \prec e_2 \prec e_3$ and c is essential, then $e_1 \not<_c e_3$.*

Proof. By contradiction. Assume $e_1 \prec e_2 \prec e_3$ and $e_1 <_c e_3$ for an essential c. Since all paths from $P = N(e_1) = s(c)$ to $P' = N(e_3) = d(c)$ go through c (by essentiality), there must exist a pair $w, r \in E$ with $e_1 \preccurlyeq w <_c r \preccurlyeq e_2$ or, symmetrically, $e_2 \preccurlyeq w <_c r \preccurlyeq e_3$, depending on whether the $w <_c r$ pair occurs before or after e_2 in the chain from e_1 to e_2 to e_3. If $e_1 \preccurlyeq w <_c r \preccurlyeq e_2 \prec e_3$, then $r <_{P'} e_3$, hence $w <_P e_1$ using ($\dagger$). If $e_1 \prec e_2 \preccurlyeq w <_c r \preccurlyeq e_3$, then $e_1 <_P w$, hence $e_3 <_{P'} r$ using ($\dagger$). In both cases we obtain a contradiction. $\square$

We now assume that c is essential and that π has $\sigma_0(c) = \sigma_n(c) = \varepsilon$ (hence E has the same number, say m, of events reading from c and writing to it). Write P for $s(c)$ and P' for $d(c)$. Let $w_1 <_P w_2 \ldots <_P w_m$ be the m events that write to c, listed in causal order. Let $r_1 <_{P'} e_2 \ldots <_{P'} r_m$ be the m events that read from c listed in causal order.

Lemma 4.4. *There exists a linear extension of $(E, \preccurlyeq)$ where, for $i = 1, \ldots, m$, w_i occurs just before r_i.*

Proof. The linear extension is constructed incrementally. Formally, for $i = 1, \ldots, m$, let $E_i \stackrel{\text{def}}{=} \downarrow r_i$ and $F_i \stackrel{\text{def}}{=} E_i \smallsetminus \{w_i, r_i\}$. Observe that $F_1 \subsetneq E_1 \subseteq F_2 \cdots F_i \subsetneq E_i \subseteq F_{i+1}$, with the convention that $F_{m+1} = E$. Every E_i is a $\preccurlyeq$-closed subset of E, also called a down-cut of $(E, \preccurlyeq)$. Furthermore, F_i is a down-cut of E_i by Lemma 4.3. Hence a linear extension of F_i followed by $w_i.r_i$ gives a linear extension of E_i, and following it with a linear extension of $F_{i+1} \smallsetminus E_i$ gives a linear extension of F_{i+1}. Any linear extension of $F_{i+1} \smallsetminus E_i$ can be chosen since this subset does not contain reads from, or writes to, c. $\square$

The linear extension we just built gives rise to a run π' in which c is synchronous. This concludes the proof of Proposition 4.2.

Observe that when several channels are essential in T, it is in general not possible to replace a run π with an equivalent π' where all essential channels are simultaneously synchronous.

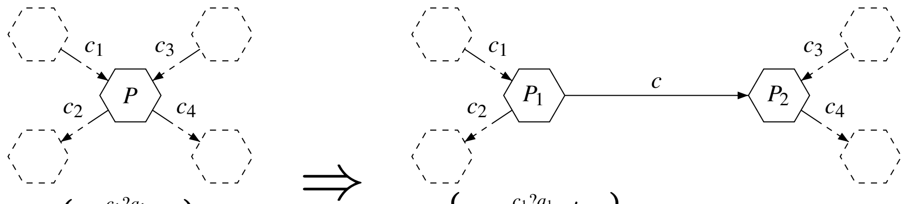
4.2 Decidability by fusion

We call "*fusion*" the transformation of T to T/c where c is essential, and "*reliable fusion*" the special case where c is also a reliable channel.

Theorem 4.5 (Decidability by fusion). *Let c be an essential channel in T:*
1. T has decidable reachability if T/c has.
2. If c is a reliable channel, then T/c has decidable reachability if T has.

Proof. 1. Let S be a T-MCS. We replace it by a system S' where c has been removed and where the processes at nodes $P_1 = s(c)$ and $P_2 = d(c)$ have been replaced by a larger process that simulate both P_1 and P_2 and where communication along c is replaced by synchronizing the sends in P_1 with the reads in P_2 (message losses are simulated even more simply by the P_1 part). S' has topology T/c and simulates S *restricted to runs where c is synchronous*. By Proposition 4.2, this is sufficient to reach any reachable configuration. Since reachability in S' is decidable, we conclude that reachability in S is decidable.

2. We now also assume that c is reliable and consider a (T/c)-MCS S. With S we associate a T-MCS S' that simulates S. S' has two nodes P_1 and P_2 where S only had a merged P node.



$$\Delta_P = \left\{ \begin{array}{c} p_1 \xrightarrow{c_1?a_1} p'_1 \\ p_2 \xrightarrow{c_2!a_2} p'_2 \\ p_3 \xrightarrow{c_3?a_3} p'_3 \\ p_4 \xrightarrow{c_4!a_4} p'_4 \\ \ldots \end{array} \right\} \qquad \Delta_{P_1} = \left\{ \begin{array}{c} p_1 \xrightarrow{c_1?a_1} p'_1 \\ p_2 \xrightarrow{c_2!a_2} p'_2 \\ p_3 \xrightarrow{c!\langle c_3,?,a_3\rangle} p'_3 \\ p_4 \xrightarrow{c!\langle c_4,!,a_4\rangle} p'_4 \\ \ldots \end{array} \right\} \quad \Delta_{P_2} = \left\{ \begin{array}{c} * \xrightarrow{c?\langle c_3,?,a_3\rangle \;\; c_3?a_3} * \\ * \xrightarrow{c?\langle c_4,?,a_4\rangle \;\; c_4?a_4} * \\ \ldots \end{array} \right\}$$

Fig. 3. Associating a T-MCS with a T/c-MCS

The construction is illustrated in Fig. 3. Informally, P_1 inherits states from P and all rules that read from channels c_1 with $d(c_1) = P_1$ in T, or write to channels c_2 with $s(c_2) = P_1$. Regarding the other rules, the communication action (reading from some c_3 or writing to some c_4) is sent to P_2 via c. S' uses an extended alphabet M' that extends the message alphabet M from S via $M' \stackrel{\text{def}}{=} M \cup (C \times \{?,!\} \times M)$. P_2 only has simple loops around a central state $*$ that read communication instructions from P_1 via c and carry them out.

S' simulates S in a strong way. Any step in S can be simulated in S', perhaps by two consecutive steps if a communication operation has to transit from P_1 to P_2 via c. In the other direction, there are some runs in S' that cannot be simulated directly by S, e.g., when P_2 does not carry out the instructions sent by P_1 (or carries them out with a delay). But all runs in S' in which c is synchronous are simulated by S.

Since runs in which c is synchronous are sufficient to reach any configuration reachable in S' (Proposition 4.2), the two-way simulation reduces reachability in S to reachability in S', which is decidable if T has decidable reachability. □

The usefulness of Theorem 4.5 is illustrated by the following two corollaries.

Corollary 4.6. *T_1^{ring} and T_3^{ring} (from Section 2.1) have decidable reachability. T_2^{ring} does not.*

Proof. Building $T_1^{ring}/c_3/c_4/c_5/c_6/c_1$ only fuses essential channels and ends up with a decidable topology (only lossy channels).
Starting with T_2^{ring}, we can build $T = T_2^{ring}/c_3/c_4/c_5/c_6$ but have to stop there (c_1 is not essential). The resulting T, isomorphic to T_4^u from Fig. 2, does not have decidable reachability. Hence T_2^{ring} does not have decidable reachability since we fused reliable channels only.
With T_3^{ring}, it is better to build $T_3^{ring}/c_3/c_4/c_6/c_1$. Here too we cannot fuse any more because of c'_2, but the end result is a topology with decidable reachability since c_5 is lossy. Hence T_3^{ring} has decidable reachability. □

Corollary 4.7. *A topology in the form of an undirected forest has decidable reachability.*

Proof (Sketch). If T is a forest, every channel c is essential, and every T/c is still a forest. Hence T reduces to a topology with lossy channels only. □

5 Splitting along lossy channels

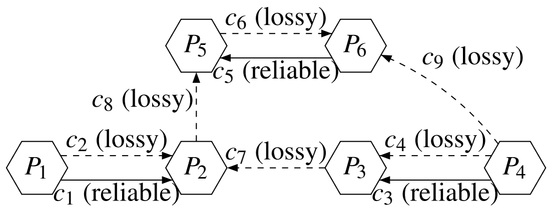


Fig. 4. A topology that splits in three

Let $T_1 = \langle N_1, R_1, L_1, s_1, d_1\rangle$ and $T_2 = \langle N_2, R_2, L_2, s_2, d_2\rangle$ be two disjoint topologies. We say that $T = \langle N, R, L, s, d\rangle$ is a *(lossy) gluing of* T_1 *on* T_2 if T is a juxtaposition of T_1 and T_2 (hence $N = N_1 \cup N_2$) with an additional set L_3 of lossy channels (hence $R = R_1 \cup R_2$ and $L = L_1 \cup L_2 \cup L_3$) connecting from T_1 to T_2 in a unidirectional way: $s(L_3) \subseteq N_1$ and $d(L_3) \subseteq N_2$.

This situation is written informally "$T = T_1 \rhd T_2$", omitting details on L_3 and its connections. In practice this notion is used to split a large T into subparts rather than build larger topologies out of T_1 and T_2.

Theorem 5.1 (Decidability by splitting). *Reachability is decidable for $T_1 \rhd T2$ if, and only if, it is for both T_1 and T_2.*

The proof of Theorem 5.1 (omitted here, see full version of this paper) uses techniques that are standard for LCS's but that have to be adapted to the more general setting of MCS's.

We can apply Theorem 5.1 to prove that the topology in Fig. 4 has decidable reachability. Indeed, this topology can be split along lossy channels (first $\{c_8, c_9\}$, then c_7), giving rise to two copies of T_2^d (from Fig. 1) and a two-node ring that can be reduced to T_1^d by fusion.

6 A complete classification

In this section, we prove that the results from the previous sections provide a complete classification.

Theorem 6.1 (Completeness). *A network topology T has decidable reachability if, and only if, it can be reduced to T_2^d (from Fig. 1) and LCS's using fusion and splitting only.*[2]

Note that, via splitting, the reduction above usually transforms T into *several* topologies. All of them must be T_2^d or LCS's for T to have decidable reachability.

The "$\Leftarrow$" direction is immediate in view of Theorems 4.5.1 and 5.1,

For the "$\Rightarrow$" direction, we can assume w.l.o.g. that T is *reduced*, i.e., it cannot be split as some $T_1 \rhd T_2$, and it does not contain any reliable essential channel (that could be fused).

We now assume, by way of contradiction, that T cannot be transformed, via general fusions, to T_2^d or to a LCS. From this we show that reachability is not decidable for T. When showing this, we sometimes mention three additional transformations ("simplification", "doubling of loops" and "non-essential fusion") that are described in the full version of this paper. We now start an involved case analysis.

1. Since T cannot be transformed to a LCS, it contains a reliable channel c_r, linking node $A = s(c_r)$ to node $B = d(c_r)$. We can assume $A \neq B$, otherwise T contains T_1^u (from Fig. 2) and we conclude immediately with undecidability.

2. T must contain a path θ of the form $A = P_0, c_1, P_1, c_2, \ldots, c_n, P_n = B$ that links A to B without using c_r, otherwise c_r would be essential, contradicting the assumption that T is reduced. We pick the shortest such θ (it is a simple path) and we call T' the subgraph of T that only contains θ, c_r, and the nodes to which they connect.

3. If all c_i's along θ are reliable, T' can be transformed to T_2^u (from Fig. 2) by reliable fusions, hence T', and then T itself, have undecidable reachability. Therefore we can assume that at least one c_i along θ is lossy.

4. Assume that there exist two nodes P_i, P_j along θ that are connected via a third path θ' disjoint from c_r and θ. We put no restrictions on the relative positions of P_i and P_j but we assume that θ' is not a trivial empty path if $i = j$. In that case, let T'' be the subgraph of T that contains c_r, θ, and θ', and where all channels except c_r are downgraded to lossy if they were reliable. Using simplification and doubling of lossy loops, T'' can

[2] As is well-known, it is possible to further reduce any LCS into T_1^d. However, we preferred a statement for Theorem 6.1 where only our two main transformations are involved.

be transformed to an undecidable topology among $\{T_3^u, T_4^u, T_5^u, T_6^u\}$. Hence T'' does not have decidable reachability. Neither has T since taking subgraphs and downgrading channels can only improve decidability.

5. If we are not in case 4, the nodes along θ do not admit a third path like θ'. Therefore all channels along θ must be lossy, since we assumed T is reduced. Thus T' can be transformed to T_2^d by general fusion. Since we assumed T cannot be transformed to T_2^d, T must contain extra nodes or channels beyond those of T'. In particular, this must include extra nodes since we just assumed that T has no third path θ' between the T' nodes. Furthermore these extra nodes must be connected to the T' part otherwise splitting T would be possible. There are now several cases.

6. We first consider the case of an extra node C with a reliable channel c from C to T'. Since T is reduced, c is not essential and there must be a second path θ' from C to T'. Call T'' the subgraph of T that only contains T', C, c and θ'. Applying non-essential fusion on c, θ' becomes a path between some P_i, P_j and we are back to case 4. Hence undecidability.

7. Next is the case of an extra node C with a reliable channel c from T' to C. Again, since c is not essential, there must be a second path θ' from T' to C. Again, the induced subgraph T'' can be shown undecidable as in case 6, reducing to case 4.

8. If there is no extra node linked to T' via a reliable c, the extra nodes must be linked to T' via lossy channels. Now the connection must go both ways, otherwise splitting would be possible. The simplest case is an extra node C with a lossy c from C to T' and a lossy c' from T' to C. But this would have been covered in case 4.

9. Finally there must be at least two extra nodes C and C', with a lossy channel c from C to T' and a lossy c' from T' to C'. We can assume that all paths between T' and C, C' go through c and c', otherwise we would be in one of the cases we already considered. Furthermore C and C' must be connected otherwise T could be split. There are several possibilities here.

10. If there is a path from C' to C we are back to case 4. Hence undecidability.

11. Thus all paths connecting C and C' go from C to C'. If one such path is made of reliable channels only, reliable fusion can be applied on the induced subgraph, merging C and C' and leading to case 8 where undecidability has been shown. If they all contain one lossy channel, T can be split, contradicting our assumption. that it is reduced.

We have now covered all possibilities when T is reduced but cannot be transformed to a LCS or to T_2^d. In all cases is has been shown that reachability is not decidable for T. This concludes the proof of Theorem 6.1.

7 A classification algorithm

Theorem 7.1 (Polynomial-time classification). *There exists a polynomial-time algorithm that classifies topologies according to whether they have decidable reachability.*

The algorithm relies on Theorem 6.1:

Stage 1: Starting from a topology T, apply splitting and *reliable* fusion as much as possible. When several transformations are possible, pick any of them nondeterministically. At any step, the transformation reduces the size of the topologies at hand, hence termination is guaranteed in a linear number of steps. At this stage we preserved decidability in both directions, hence T has decidability iff all the reduced topologies $T_1, \dots, T_n$ have.

Stage 2: Each T_i is now simplified using general fusion (not just reliable fusion). If this ends with a LCS or with T_2^d, decidability for T_i has been proved. When fusion can be applied in several ways, we pick one nondeterministically: a consequence of Theorem 6.1's proof is that these choices lead to the same conclusion when starting from a system that cannot be reduced with splitting or reliable fusion. Thus stage 2 terminates in a linear number of steps. When it terminates, either every T_i has been transformed into a LCS or T_2^d, and we conclude that reachability is decidable for T, or one T_i remains unsimplified and we conclude that reachability is not decidable for T.

We observe that when stage 1 finishes, there will never be any new opportunity for reliable fusion or for splitting since stage 2, i.e., general fusion, does not create or destroy any path between nodes.

8 Concluding remarks

Summary. We introduced *mixed channel systems*, i.e., fifo channel systems where both lossy and reliable channels can be combined in arbitrary topologies. These systems are a generalization of the lossy channel system model (where all channels are lossy and where reachability is decidable) and of the standard model (with unbounded reliable fifo channels, where reachability is undecidable).

For mixed systems, we provide a complete classification of the network topologies according to whether they lead to decidable reachability problems or not. Our main tool are reductions methods that transform a topology into simpler topologies with an equivalent decidability status. These reductions produce small basic topologies for which the decidability status is established in Section 3.

Directions for future work. At the moment our classification is given implicitly, via a simplification procedure. A more satisfactory classification would be a higher-level description, in the form of a structural criterion, preferably expressible in logical form (or via excluded minors, ...). Obtaining such a description is our more pressing objective.

Beyond this issue, the two main avenues for future work are extending the MCS model (e.g., by considering other kinds of unreliability in the style of [9], or by allowing guards in the style of [5], etc.) and considering questions beyond just reachability and safety (e.g., termination and liveness).

References

1. P. A. Abdulla, C. Baier, S. Purushothaman Iyer, and B. Jonsson. Simulating perfect channels with probabilistic lossy channels. *Information and Computation*, 197(1–2):22–40, 2005.
2. P. A. Abdulla, N. Bertrand, A. Rabinovich, and Ph Schnoebelen. Verification of probabilistic systems with faulty communication. *Information and Computation*, 202(2):141–165, 2005.

3. P. A. Abdulla, A. Collomb-Annichini, A. Bouajjani, and B. Jonsson. Using forward reachability analysis for verification of lossy channel systems. *Formal Methods in System Design*, 25(1):39–65, 2004.
4. P. A. Abdulla and B. Jonsson. Verifying programs with unreliable channels. *Information and Computation*, 127(2):91–101, 1996.
5. C. Baier, N. Bertrand, and Ph. Schnoebelen. On computing fixpoints in well-structured regular model checking, with applications to lossy channel systems. In *Proc. LPAR 2006*, volume 4246 of *Lecture Notes in Artificial Intelligence*, pages 347–361. Springer, 2006.
6. C. Baier, N. Bertrand, and Ph. Schnoebelen. Verifying nondeterministic probabilistic channel systems against ω-regular linear-time properties. *ACM Trans. Computational Logic*, 9(1), 2007.
7. P. Bouyer, N. Markey, J. Ouaknine, Ph. Schnoebelen, and J. Worrell. On termination for faulty channel machines. In *Proc. STACS 2008*, pages 121–132, 2008.
8. D. Brand and P. Zafiropulo. On communicating finite-state machines. *Journal of the ACM*, 30(2):323–342, 1983.
9. 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.
10. P. Chambart and Ph. Schnoebelen. Post embedding problem is not primitive recursive, with applications to channel systems. In *Proc. FST&TCS 2007*, volume 4855 of *Lecture Notes in Computer Science*, pages 265–276. Springer, 2007.
11. P. Chambart and Ph. Schnoebelen. The ω-regular Post embedding problem. In *Proc. FOSSACS 2008*, volume 4962 of *Lecture Notes in Computer Science*, pages 97–111. Springer, 2008.
12. P. Chambart and Ph. Schnoebelen. The ordinal recursive complexity of lossy channel systems. In *Proc. LICS 2008*. IEEE Comp. Soc. Press, 2008.
13. A. Finkel. Decidability of the termination problem for completely specificied protocols. *Distributed Computing*, 7(3):129–135, 1994.
14. J. G. Henriksen, M. Mukund, K. N. Kumar, M. A. Sohoni, and P. S. Thiagarajan. A theory of regular MSC languages. *Information and Computation*, 202(1):1–38, 2005.
15. M. Jurdziński and R. Lazić. Alternation-free modal mu-calculus for data trees. In *Proc. LICS 2007*, pages 131–140. IEEE Comp. Soc. Press, 2007.
16. A. Kurucz. Combining modal logics. In P. Blackburn, J. van Benthem, and F. Wolter, editors, *Handbook of Modal Logics*, volume 3, chapter 15, pages 869–926. Elsevier Science, 2006.
17. S. Lasota and I. Walukiewicz. Alternating timed automata. *ACM Trans. Computational Logic*, 9(2), 2008.
18. M. Lohrey and A. Muscholl. Bounded MSC communication. *Information and Computation*, 189(2):160–181, 2004.
19. J. Ouaknine and J. Worrell. On the decidability and complexity of Metric Temporal Logic over finite words. *Logical Methods in Comp. Science*, 3(1):1–27, 2007.
20. J. K. Pachl. Protocol description and analysis based on a state transition model with channel expressions. In *Proc. PSTV '87*, pages 207–219. North-Holland, 1987.
21. Ph. Schnoebelen. The verification of probabilistic lossy channel systems. In C. Baier et al., editors, *Validation of Stochastic Systems – A Guide to Current Research*, volume 2925 of *Lecture Notes in Computer Science*, pages 445–465. Springer, 2004.