Decidable Topologies for Communicating Automata with FIFO and Bag Channels[⋆]

Lorenzo Clemente, Frédéric Herbreteau, and Grégoire Sutre

Univ. Bordeaux, CNRS, LaBRI, UMR 5800, Talence, France.

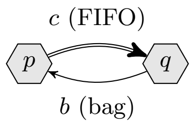
Abstract. We study the reachability problem for networks of finite-state automata communicating over unbounded perfect channels. We consider communication topologies comprising both ordinary FIFO channels and *bag channels*, i.e., channels where messages can be freely reordered. It is well-known that when only FIFO channels are considered, the reachability problem is decidable if, and only if, there is no undirected cycle in the topology. On the other side, when only bag channels are allowed, the reachability problem is decidable for any topology by a simple reduction to Petri nets. In this paper, we study the more complex case where the topology contains *both* FIFO and bag channels, and we provide a complete characterisation of the decidable topologies in this generalised setting. One consequence of our results is that, in presence of both kind of channels, certain non-trivial cycles can be allowed while preserving decidability.

1 Introduction

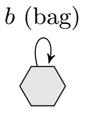
Communicating finite-state automata are a fundamental model of computation where concurrent processes exchange messages over unbounded FIFO channels [2,16]. However, the model is Turing-powerful, and even basic verification questions, like reachability, are undecidable. To obtain decidability, various restrictions have been considered in the literature, including making channels unreliable [1,4,5,7], restricting to half-duplex communication [3] (later generalised to mutex [8]), bounded context-switching [11], or constraining the communication topology by forbidding cycles [16]. Here, we study the reachability problem for topologies mixing FIFO and *bag channels* (a.k.a. unordered or multi-set channels).

Motivation. Models with only FIFO channels or only bag channels are well-understood. On the one hand, topologies with only FIFO channels are well-known to be decidable exactly when the communication topology does not contain any undirected cycle [16,11]. On the other hand, topologies with only bag channels are much easier to analyse: They are always decidable, independently of the topology, by a simple reduction to Petri nets, for which reachability is decidable [14,9,12]. While certainly bag channels have been already considered before as a form of asynchronous communication (cf. the task-based approach in [18]), we are

⋆ This work was partially supported by the ANR project Vacsim (ANR-11-INSE-004).



(a) Opposite channels b and c.



(b) Fusing c

Fig. 1: Two opposite channels

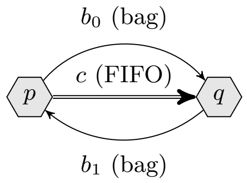


Fig. 2: Undecidable topology

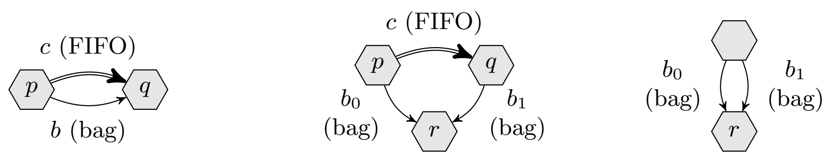
not aware of any work studying the reachability problem for mixed topologies comprising *both* bag and FIFO channels. Our complete characterisation result provides a solution to it.

There is also a practical interest in mixing FIFO and bag channels. Bag channels provide an over-approximation of FIFO channels which is easier to analyse. As we solve the reachability problem, if an error configuration can be reached in a model with FIFO channels, then so it is the case if some FIFO channels are replaced by bag channels. While the reachability problem might be undecidable in the original topology, it may be decidable in the abstracted one. Till now, there was no other choice than replacing *all* FIFO channels by bag channels to gain decidability. Our characterisation suggests that it suffices to selectively weaken only some of the FIFO channels to obtain a decidable topology, yielding much refined abstractions for the verification of communicating systems.

Another motivation is that bag channels can be used to implement *acknowledgements* in communication protocols. Consider a producer process p that sends messages to a consumer process q. Suppose we are interested in an implementation of the protocol where the channel from p to q never contains more than one message. This can be achieved by introducing a reverse bag channel from q to p on which q sends acknowledgements each time a message is received, and p only sends the next message when the previous one has been acknowledged. More generally, this technique can be used to model sliding window protocols that control the relative speed of the processes.

Our contributions. We provide a complete characterisation of decidable topologies comprising both FIFO and bag channels. Let us illustrate our main techniques with some example. While the topology in Fig. 1a is undecidable in the FIFO setting we show that when one of the channels (say b) is a bag channel, reachability becomes decidable. Indeed, the FIFO channel c can be bounded to size at most one, and it can in fact be removed by performing a synchronised product of p and q. In the resulting topology, b becomes a self-loop bag channel (cf. Fig. 1b). As we have observed above, this is equivalent to a Petri net, and thus decidable.

A more difficult case is the one in Fig. 3a. As above, reachability is undecidable if both channels c and b are FIFO, and it becomes decidable as soon as one channel (say b) is a bag channel. However, the correctness argument is more involved here, since, unlike in the previous example, channel c cannot be bounded. The problem is that bounding c requires rescheduling the actions of the receiver q to occur



(a) Parallel channels b and c (b) Splitting b (c) Fusing c

Fig. 3: Two parallel channels

earlier. But this is not possible since q might try to read on the other channel b, which could be empty (for example). However, the crucial observation is that we can always schedule all actions of p first (since p only sends messages), followed by all actions of q. Therefore, in this topology, the order between transmissions and receptions can be relaxed. The only thing that matters is that the string of messages which is received is the same as the one which is sent. We can thus *split* the bag channel $p \xRightarrow{b} q$ into two bag channels $p \xRightarrow{b_0} r$ and $r \xLeftarrow{b_1} q$ (cf. Fig. 3b), where q's potentially blocking receptions on b are replaced with non-blocking transmissions on b_1, and the new process r just matches incoming messages on b_0 and b_1. In the new topology, c can be bounded, therefore we can fuse p and q to obtain the topology in Fig. 3c. Again, as we now have only bag channels, reachability is decidable by reduction to Petri nets.

We also have undecidable topologies which mix in a non-trivial way FIFO and bag channels. For example, consider the one in Fig. 2, where c is FIFO and b_0 and b_1 are bags. This topology is undecidable, even when b_0 and b_1 are unary bag channels (i.e., the message alphabet is a singleton). The idea is to use the two bag channels b_0 and b_1 to implement a synchronisation protocol between processes p and q. This protocol is then used by p to force which message is received by q from c, thus simulating a channel machine (which has undecidable reachability).

Finally, bag channels can be used to simulate tick automata communicating over FIFO channels [6]. Communicating tick automata provide a model of communication in discrete time where synchronisation occurs over a global rendezvous action τ. A useful observation is that synchronisation can be relaxed if we allow the receiver q to keep track of how many time units it is running ahead of the sender p [10]. While this information can be stored in a non-negative counter in q [6], we give a simpler construction by just adding an acknowledgement bag channel (even unary) from q to p, and having p and q receive/send acknowledgements instead of performing rendezvous actions τ. Our characterisation result generalises that for communicating tick automata in [6].

Outline. The rest of the paper is organised as follows. We start with technical preliminaries in Sec. 2. Then in Sec. 3 we show basic techniques for fusing and splitting channels, in order to reduce decidable topologies to simpler ones, and in Sec. 4 we explain how unary bag channels can be used to simulate rendezvous synchronisation (for undecidability). In Sec. 5 we state and prove our characterisation result of decidable topologies for communicating automata with FIFO and bag channels, by using the techniques of the two previous sections, and in Sec. 6 we generalise the characterisation to discrete-time communicating tick automata. Finally, in Sec. 7 we compare our results and techniques with the work of Chambart and Schnoebelen [5], and in Sec. 8 we end with further work. Full proofs can be found in Appendices A, B, and C.

2 Preliminaries

A *labelled transition system* (LTS for short) is a tuple $\mathcal{A} = \langle S, S_I, S_F, A, \to\rangle$ where S is a set of *states* with *initial states* $S_I \subseteq S$ and *final states* $S_F \subseteq S$, A is a finite set of *actions*, and $\to \subseteq S \times A \times S$ is a *labelled transition relation.* For simplicity, we write $s \xrightarrow{a} s'$ in place of $(s, a, s') \in \to$. A LTS is finite when S is finite. A *run* in $\mathcal{A}$ is an alternating sequence $\pi = s_0, a_1, s_1, \ldots, a_n, s_n$ of states $s_i \in S$ and actions $a_i \in A$ s.t. $s_{i-1} \xrightarrow{a_i} s_i$ for all $i \in \{1, \ldots, n\}$. An *accepting run* is a run starting in an initial state ($s_0 \in S_I$) and ending in a final state ($s_n \in S_F$).

Topologies. We consider systems that are composed of several processes communicating through the asynchronous exchange of messages. This is modelled by introducing FIFO and bag channels between processes. While a FIFO channel preserves the order of messages, bag channels only keep the number of messages in the channel for each type. Formally, a communication *topology* is a tuple $\mathcal{T} = \langle P, C, B, \mathsf{src}, \mathsf{dst}\rangle$, where P is a finite set of *processes*, C is a finite set of *channels*, of which those in $B \subseteq C$ are *bag channels* and those in $C\backslash B$ are *FIFO channels*, and src and $\mathsf{dst} : C \mapsto P$ are mappings assigning to each channel a *source* and a *destination* process.

We write $p \overset{c}{\Longrightarrow} q$ when c is a channel with $\mathsf{src}(c) = p$ and $\mathsf{dst}(c) = q$, and we write $p \overset{c}{\Longleftrightarrow} q$ when there is a channel either way, i.e., $p \overset{c}{\Longrightarrow} q$ or $q \overset{c}{\Longrightarrow} p$. An *undirected path* is an alternating sequence $p_0, c_1, p_1, \ldots, c_n, p_n$ of processes and channels such that:

$$p_0 \overset{c_1}{\Longleftrightarrow} p_1 \overset{c_2}{\Longleftrightarrow} \cdots \overset{c_n}{\Longleftrightarrow} p_n$$

by default, a path is undirected

A path is *directed* if $\mathsf{dst}(c_i) = \mathsf{src}(c_{i+1})$ for all $1 \leqslant i < n$. The notation $p \overset{*}{\Longrightarrow} q$ means that there is a directed path from p to q. A *cycle* is a path that starts and ends in the same process: $p_n = p_0$. A path is *simple* if all channels and all processes are distinct. Similarly a cycle is *simple* if all channels and all processes, except p_0 and p_n, are distinct. The *support* of a path is the set $D = \{c_1, c_2, \ldots, c_n\}$ of channels that it visits.

Communicating processes. Given a topology $\mathcal{T}$ as above and a finite set M of messages, the set of possible *communication actions* for process $p \in P$ is $A^p_{\text{com}} = \{c!m \mid \mathsf{src}(c) = p, m \in M\} \cup \{c?m \mid \mathsf{dst}(c) = p, m \in M\}$. The set of all communication actions is $A_{\text{com}} = \bigcup_{p\in P} A^p_{\text{com}}$. Actions not in A_{com} are called *internal actions.*

Definition 2.1. *A* system of communicating processes *is a tuple* $\mathcal{S} = \langle \mathcal{T}, M, (\mathcal{A}^p)_{p\in P}\rangle$ *where* $\mathcal{T}$ *is a topology,* M *is a finite set of* messages*, and, for each* $p \in P$, $\mathcal{A}^p = \langle S^p, S^p_I, S^p_F, A^p, \rightarrow^p\rangle$ *is a finite labelled transition system s.t.* $A^p \cap A_{\mathrm{com}} = A^p_{\mathrm{com}}$.

States $s^p \in S^p$ are called *local states* of p, while a *global state* $\boldsymbol{s} = (s^p)_{p\in P}$ is a tuple of local states in $\prod_{p\in P} S^p$. We give the semantics of a system of communicating processes in terms of a global labelled transition system. Formally, the *semantics of a system of communicating processes* $\mathcal{S} = \langle \mathcal{T}, M, (\mathcal{A}^p)_{p\in P}\rangle$ is the labelled transition system $[\![\mathcal{S}]\!] = \langle S, S_I, S_F, A, \rightarrow\rangle$ where $S = (\prod_{p\in P} S^p)\times(M^*)^C$, $S_I = (\prod_{p\in P} S^p_I) \times \{\lambda c\,.\,\varepsilon\}$, $S_F = (\prod_{p\in P} S^p_F) \times \{\lambda c\,.\,\varepsilon\}$, $A = \bigcup_{p\in P} A^p$, and there is a transition $(\boldsymbol{s}_1, w_1) \xrightarrow{a} (\boldsymbol{s}_2, w_2)$ if there exists a process $p \in P$ such that:

– $s^p_1 \xrightarrow{a} s^p_2$ is a transition in $\mathcal{A}^p$ and for all other processes $q \in P\backslash\{p\}$, $s^q_1 = s^q_2$.
– If a is an internal action, then $w_1 = w_2$. Otherwise:
  - If $a = c!m$, then $w_2(c) = w_1(c) \cdot m$.
  - If $a = c?m$ and $c \in C\backslash B$ is a FIFO channel, then $w_1(c) = m \cdot w_2(c)$.
  - If $a = c?m$ and $c \in B$ is a bag channel, then $w_1(c) = u \cdot m \cdot v$ and $w_2(c) = u \cdot v$ for some $u, v \in M^*$.

  And all other channels $d \in C\backslash\{c\}$ are left untouched: $w_2(d) = w_1(d)$.

For uniformity of notations, we represent bag channels as words which are however interpreted as multisets. Indeed a message can be received from any position in the channel. Notice that a bag channel is equivalent to a bunch of unary channels in parallel (one channel per message type), where a unary channel is just a channel over a one-message alphabet [17].

Two runs π and ρ are *order-equivalent* if they can be transformed one into the other by iteratively commuting adjacent actions that (1) are not on the same process and (2) do not contradict causality of communications.

REVIEWS 1&2&3: Define this notion more formally.

Statement of the problem. Given a topology $\mathcal{T}$, the *reachability problem* for systems of communicating processes with topology $\mathcal{T}$, denoted by REACH($\mathcal{T}$), is defined as follows:

Input: a system of communicating processes $\mathcal{S}$ with topology $\mathcal{T}$,
Output: whether there exists an accepting run in $[\![\mathcal{S}]\!]$.

Observe that we require all channels to be empty at the end of an accepting run. Also note that REACH($\mathcal{T}$) is parametrized by a topology $\mathcal{T}$. The main result of this paper is a characterisation of topologies $\mathcal{T}$ for which REACH($\mathcal{T}$) is decidable. Our techniques consist in certain topological transformations which induce reducibility in the associate reachability problems. Formally, given two topologies $\mathcal{T}$ and $\mathcal{U}$, $\mathcal{T} \sqsubseteq \mathcal{U}$ if REACH($\mathcal{T}$) is reducible to REACH($\mathcal{U}$).For example, $\mathcal{T} \sqsubseteq \mathcal{U}$ holds when $\mathcal{U}$ is obtained from $\mathcal{T}$ by identifying and merging two processes.

REVIEW 1: be more formal about our notion of reducibility; logspace many-one reductions?.

The following characterisation for REACH($\mathcal{T}$) is well-known when the topology contains only FIFO channels.

Theorem 2.2 ([16]). *Given a topology* $\mathcal{T}$ *with no bag channel,* REACH($\mathcal{T}$) *is decidable if, and only if,* $\mathcal{T}$ *has no undirected cycle.*

In Sec. 5 we generalise this condition to our more general setting comprising both FIFO and bag channels (cf. Theorem 5.2), and in Sec. 6 we further generalise it to communicating tick automata (cf. Theorem 6.1). Our undecidability results still hold when bag channels are replaced by unary channels.

3 Fusing and splitting channels

A useful technique in the analysis of communicating automata is to bound the length of channels without compromising the behaviour of the system. For example, it is well-known that in a polyforest topology (i.e., without undirected cycles) any run can be reordered to have *all* channels bounded to length at most one (existentially 1-bounded channels [13]), and reachability can be decided by exploring the resulting finite transition system. Since we are interested in analysing more complex topologies where not all channels can be simultaneously bounded, we need to be able to bound channels individually rather than globally.

To this end, we use the notion of synchronous runs from [5,8]. Formally, a run $\pi = ((\boldsymbol{s}_0, w_0), a_1, (\boldsymbol{s}_1, w_1), \ldots, a_n, (\boldsymbol{s}_n, w_n))$ is *synchronous* for a channel c if $w_0(c) = w_n(c) = \varepsilon$ and $w_i(c) = \varepsilon \vee w_{i+1}(c) = \varepsilon$ for all $1 \leqslant i < n$. Intuitively, this condition requires that communication on c behaves like rendezvous synchronisation [15]. If every accepting run can be reordered into a run that is synchronous for c, then c can be removed altogether by merging the sender and the receiver without impacting accepting runs.

Fusing essential channels. Whether a channel can be made synchronous (via a reordering of actions in the run) is a semantic condition which depends on the complex behaviour of the whole system. In fact, this condition is an undecidable problem in general (by an immediate reduction from the reachability problem for FIFO automata). Therefore, we are interested in syntactic conditions that are sufficient for a channel to be made synchronous. One such condition is that of *essential channel* [5], which is a structural condition depending only on the topology.

Definition 3.1 ([5]). *A channel $p \overset{c}{\Longrightarrow} q$ is* essential *if all directed paths from p to q contain c.*

Lemma 3.2 ([5]). *If c is an essential channel, then every run starting and ending with c empty is order-equivalent to a run that is synchronous for c.*

Proof. The proof given in [5] for communicating processes with mixed lossy and reliable (FIFO) channels can be adapted to our setting comprising mixed bag and FIFO channels. An alternative, direct proof is given in Appendix A. □

Therefore, a system $\mathcal{S}$ whose topology contains an essential channel c can be simulated by a simpler system $\mathcal{S}'$ where c is removed and p and q are merged into a single process. Intuitively, communication on c in $\mathcal{S}$ is replaced by rendezvous synchronisation in $\mathcal{S}'$. We obtain the following proposition. Given a topology $\mathcal{T}$ and a channel $p \overset{c}{\Longrightarrow} q$, the *fusion* of c in $\mathcal{T}$ is the topology obtained from $\mathcal{T}$ by removing c and merging p and q [5]. All other channels involving p or q are redirected in the obvious way.

REVIEW 1: add a more formal definition for fusion?

Proposition 3.3 (Fusion). *If c is an essential channel in a topology $\mathcal{T}$, then $\mathcal{T} \sqsubseteq \mathcal{U}$ where $\mathcal{U}$ results from the fusion of c in $\mathcal{T}$.*

Splitting irreversible channels. According to the proposition above, we can always reduce to a smaller topology by fusing an essential channel. However, there are situations where a channel is not essential, but it can be made so after a small modification. Consider the (decidable) topology in Fig. 3a on page 3. Neither channel c nor b is essential, or can be bounded in general. However, we observe that c can be restricted to length at most one if we allow q to receive messages from b that have not been sent yet. This is correct since, in this topology, we can always schedule all actions of p to occur before any action of q. Therefore, what only matters here is that the whole sequence of messages sent by p is received by q, and causality between transmissions and receptions can in fact be relaxed. We relax causality in $p \overset{b}{\Longrightarrow} q$ by splitting it into two channels $p \overset{b_0}{\Longrightarrow} r$ and $r \overset{b_1}{\Longleftarrow} q$ (see Fig. 3b on page 3), where the new process r just matches messages on both channels; below, we show that the latter topology simulates the former. Notice that channel c becomes essential after the splitting of b.

We now make these ideas more precise. Given a topology $\mathcal{T}$ and a channel $p \overset{c}{\Longrightarrow} q$, the *split* of c in $\mathcal{T}$ is the topology obtained from $\mathcal{T}$ by adding a new process r and splitting the channel $p \overset{c}{\Longrightarrow} q$ into two channels $p \overset{c_0}{\Longrightarrow} r$ and $r \overset{c_1}{\Longleftarrow} q$. Moreover, c_0 and c_1 are bag channels if c is a bag channel, and c_0 and c_1 are FIFO channels if c is a FIFO channel.[1] Splitting a channel removes directed paths from the topology, and it might make other channels essential, which is beneficial. However, not all channels can be split. To justify this operation, we introduce the notion of causal run. A run is *causal* for a process p if every process q that moves before some move of p in the run verifies $q \overset{*}{\Longrightarrow} p$.

REVIEW 1: add formal definition.

Lemma 3.4. *Given a process p, every run is order-equivalent to a run that is causal for p.*

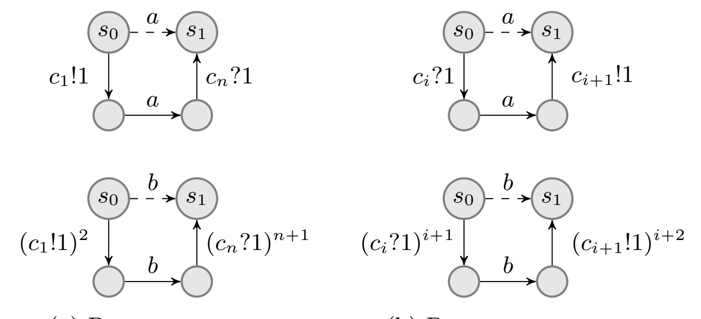
We want to ensure that, after splitting a channel $p \overset{c}{\Longrightarrow} q$ into $p \overset{c_0}{\Longrightarrow} r$ and $r \overset{c_1}{\Longleftarrow} q$, a causal run for p ensures that q is never scheduled before p in the split system. A sufficient condition is given by the notion of *irreversible channel*.

Definition 3.5. *A channel $p \overset{c}{\Longrightarrow} q$ is* irreversible *if there exists no directed path from q to p.*

Clearly, after splitting an irreversible channel there still is no directed path from q to p, and thus, by the definition of causal run for p, all moves of q necessarily follow the last move of p.

REVIEW 3: clarify "have to". made into "necessarily"

[1] Despite having similar names, this splitting notion and the splitting technique of [5] have little in common (see Sec. 7).



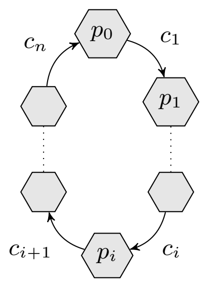


Fig. 4: Synchronisation protocol for a simple directed cycle (and a unary alphabet)

Proposition 3.6 (Split). *If c is an irreversible channel in a topology $\mathcal{T}$, then $\mathcal{T} \sqsubseteq \mathcal{U}$ where $\mathcal{U}$ results from the split of c in $\mathcal{T}$.*

4 The power of unary channels

In this section, we show that the behaviour of a single process can be distributed over a cyclic sub-topology. While simple distribution protocols exist when the cyclic sub-topology contains FIFO or bag channels over an unrestricted message alphabet, in this section we impose ourselves the much stronger constraint that all channels therein are over a *unary* message alphabet. This will yield a stronger characterisation result in Sec. 5. Thus, for the remainder of this section, we assume that all channels of the considered cyclic sub-topology are unary; let 1 denote the unique message that is exchanged over those channels.

We begin by extending the notion of fusion from Sec. 3 to sets of channels. Given a topology $\mathcal{T}$ and a set of channels $D \subseteq C$, the *fusion* of D in $\mathcal{T}$ is the topology obtained from $\mathcal{T}$ by removing D and merging, into a single process, the set $\bigcup_{c \in D}\{\mathsf{src}(c), \mathsf{dst}(c)\}$ of all endpoints of channels in D. As before, all other channels involving those endpoints which are fused are redirected in the obvious way. This section shows that fusing a cyclic sub-topology makes the reachability problem easier. We first deal with the case of simple directed cycles. Recall that the support of a directed path is the set of channels that it visits.

REVIEW 1: add a formal definition.

Lemma 4.1. *If D is the support of a simple directed cycle in a topology $\mathcal{T}$ then $\mathcal{U} \sqsubseteq \mathcal{T}$ where $\mathcal{U}$ results from the fusion of D in $\mathcal{T}$.*

Proof. Consider a simple directed cycle $p_0 \xRightarrow{c_1} p_1 \xRightarrow{c_2} \cdots \xRightarrow{c_n} p_n = p_0$ in $\mathcal{T}$ such that $D = \{c_1, \ldots, c_n\}$. This simple directed cycle is depicted in Fig. 4c. Let $\mathcal{U}$ denote the topology that results from the fusion of D in $\mathcal{T}$. Denote by u the process in $\mathcal{U}$ that corresponds to the merging of all endpoints of D. Consider a system $\mathcal{S}$ with topology $\mathcal{U}$. We construct a new system $\mathcal{S}'$ with topology $\mathcal{T}$ that simulates $\mathcal{S}$ as follows.

Our strategy is to exploit the cyclic nature of D to simulate rendezvous synchronisation[2] between the processes $p_0, \ldots, p_{n-1}$. Clearly, this allows $p_0, \ldots, p_{n-1}$ to coordinate in $\mathcal{T}$ and simulate any behaviour of u in $\mathcal{U}$. We only show how to simulate rendezvous synchronisation over two actions, call them a and b. Rendezvous synchronisation over a larger alphabet $\{c_1, \ldots, c_m\}$ can be simulated, for example, by replacing one rendezvous synchronization c_i with $i+1$ rendezvous synchronizations $a^i b$, for $1 \leqslant i \leqslant m$ (other encodings are possible).

REVIEW 2: give the idea of the encoding.

In our simulation, rendezvous synchronisations are initiated by p_0, and then propagated along the cycle $p_0 \xRightarrow{c_1} p_1 \xRightarrow{c_2} \cdots \xRightarrow{c_n} p_n$ back to p_0. The latter then checks that the other processes correctly performed the desired rendezvous action. More precisely, whenever p_0 wants to handshake on some action (a or b), it sends some number of messages on c_1 to p_1 and waits for an acknowledgement on c_n before proceeding to its next move. In the meantime, the processes $p_1, \ldots, p_{n-1}$ do the same, but first receive and then transmit. The number of messages received by p_{i+1} from c_{i+1} is exactly the same as the number of messages sent by p_i on c_{i+1}. The precise protocol is shown in Fig. 4, where each dashed transitions in $\mathcal{S}$ is replaced in $\mathcal{S}'$ by the alternative sequence below it (by introducing intermediate states). All other transitions are left unchanged. In $\mathcal{S}'$, actions a and b are to be interpreted as internal, non-rendezvous actions. For instance, to simulate the rendezvous action a, p_0 first sends a message on c_1, internally performs a, and then receives a message from c_n.

By construction, each rendezvous synchronisation in $\mathcal{S}$ can be reproduced, through the above protocol, in $\mathcal{S}'$. We now argue that the protocol does not introduce any spurious behaviour. Recall that the channels $c_1, \ldots, c_n$ are empty at the beginning. So, for each $1 \leqslant i \leqslant n-1$, the process p_i may only simulate a rendezvous action after p_0 has initiated a synchronisation round. Let us look at the first rendezvous action that is simulated by p_0. If this action is a, then p_0 sends one message on c_1 and receives one message from c_n. This entails that $p_1, \ldots, p_{n-1}$ simulate, each, the rendezvous action a. At the end of this synchronisation round, all the channels $c_1, \ldots, c_n$ are again empty. If the first rendezvous action that p_0 simulates is b, then p_0 sends two messages on c_1 and receives $n+1$ messages from c_n. Again, this entails that $p_1, \ldots, p_{n-1}$ simulate, each, the rendezvous action b. Indeed, by contradiction, if p_i simulates a, then it must continue simulating a since there are not enough messages in c_i anymore to simulate b. Therefore, it simply relays messages from c_i to c_{i+1}, and cannot produce on c_{i+1} the extra message that p_{i+1} expects to simulate b. By applying the same arguments to the remaining processes $p_{i+1}, \ldots, p_{n-1}$, we obtain that p_{n-1} is not able to produce on c_n the $n+1$ messages that p_0 expects to complete its simulate of b, a contradiction. Again, at the end of this synchronisation round, all the channels $c_1, \ldots, c_n$ are empty. By repeating this analysis for each synchronisation round, we obtain that every accepting run of $\mathcal{S}'$ can be mapped back to an accepting run of $\mathcal{S}$.

[2] Here, we implicitly consider multi-way rendezvous synchronisations between all processes $p_0, \ldots, p_{n-1}$.

While we have shown correctness for the processes $p_0, \dots, p_{n-1}$ in isolation, the protocol is local and it works properly even when those are part of the larger topology $\mathcal{T}$ and exchange messages with other processes outside D. $\square$

We now show that the previous lemma still holds for arbitrary directed cycles. The proof is by induction on the cardinality of D. As expected, the induction step follows from Lemma 4.1.

Proposition 4.2 (Synchronisation). *If D is the support of a directed cycle in a topology $\mathcal{T}$, then $\mathcal{U} \sqsubseteq \mathcal{T}$ where $\mathcal{U}$ results from the fusion of D in $\mathcal{T}$.*

REVIEW 1: the induction hypothesis has been clarified.

Proof. By induction on the cardinality of D. The basis is trivial since the fusion of $\varnothing$ in $\mathcal{T}$ is $\mathcal{T}$ itself. Let D be a non-empty set of channels, and assume that the proposition holds for all proper subsets of D which are supports of directed cycles. Denote by $\mathcal{U}$ the fusion of D in $\mathcal{T}$. By definition, there exists a directed cycle $p_0 \xRightarrow{c_1} p_1 \xRightarrow{c_2} \cdots \xRightarrow{c_n} p_n = p_0$ in $\mathcal{T}$ whose support is $D = \{c_1, \dots, c_n\}$. If the cycle is simple, then we conclude thanks to Lemma 4.1. Otherwise, there exists $1 \leqslant k \leqslant l \leqslant n$ such that $p_{k-1} \xRightarrow{c_k} p_k \xRightarrow{c_{k+1}} \cdots \xRightarrow{c_l} p_l$ is a simple directed cycle. By Lemma 4.1, it holds that $\hat{\mathcal{T}} \sqsubseteq \mathcal{T}$ where $\hat{\mathcal{T}}$ results from the fusion of $\{c_k, \dots, c_l\}$ in $\mathcal{T}$. Denote by $\hat{p}$ the process in $\hat{\mathcal{T}}$ that corresponds to the merging of all endpoints of $\{c_k, \dots, c_l\}$. Let $\hat{D} = D \backslash \{c_k, \dots, c_l\}$. The cycle $p_0 \xRightarrow{c_1} p_1 \xRightarrow{c_2} \cdots \xRightarrow{c_n} p_n = p_0$ of $\mathcal{T}$ gives rise to a cycle in $\hat{\mathcal{T}}$ with support $\hat{D}$. Indeed, the new cycle is obtained by collapsing all $p_{i-1} \xRightarrow{c_i} p_i$ with $c_i \in \{c_k, \dots, c_l\}$ into $\hat{p}$. Observe that $\mathcal{U}$ also results from the fusion of $\hat{D}$ in $\hat{\mathcal{T}}$. We derive from the induction hypothesis that $\mathcal{U} \sqsubseteq \hat{\mathcal{T}} \sqsubseteq \mathcal{T}$, which concludes the proof of the proposition. $\square$

Proposition 4.2 above will be used in the undecidability part of our characterisation result presented in the next section.

5 Characterisation

We are now ready to state and prove our characterisation of decidable topologies mixing FIFO and bag channels. The characterisation is expressed in the same vein as Theorem 2.2, and generalises it.

Definition 5.1. *Let $D \subseteq C$ be a set of channels. Two processes p and q are* synchronizable over D, *written $p \approx_D q$, if there exist directed paths from p to q, and back from q to p, using only channels in D.*

A *jumping cycle* is a sequence $p_0, c_1, q_1, p_1, \dots, c_n, q_n, p_n$ of processes p_i, q_i and channels c_i such that $c_1, \dots, c_n$ are distinct FIFO channels, and

$$p_0 \overset{c_1}{\Longleftrightarrow} q_1 \approx_D p_1 \overset{c_2}{\Longleftrightarrow} q_2 \approx_D \cdots \overset{c_n}{\Longleftrightarrow} q_n \approx_D p_n = p_0$$

where $D = C \backslash \{c_1, \dots, c_n\}$. Recall that $p \overset{c}{\Longleftrightarrow} q$ holds iff $p \xRightarrow{c} q$ or $q \xRightarrow{c} p$.

Theorem 5.2. *Given a topology $\mathcal{T}$, $\textsc{Reach}(\mathcal{T})$ is decidable if, and only if, $\mathcal{T}$ has no jumping cycle, even when bag channels are over a unary message alphabet.*

The two directions of the theorem are proved in the two subsections below. The proofs of Lemma 5.3 and Lemma 5.4 are given in Appendix B. To illustrate our characterisation result, let us give some example of decidable topologies. Certainly, polyforest topologies are decidable since they do not contain jumping cycles. Moreover, decidability is preserved if each node in the polyforest is expanded into a sub-topology of only bag channels. Even further, we still get a decidable topology if, for every essential FIFO channel $p \overset{c}{\Longrightarrow} q$, we add an acknowledgement bag channel $q \overset{c^{-1}}{\Longrightarrow} p$ (cf. Lemma B.1 in Appendix B). This operation introduces non-trivial cycles of FIFO and bag channels. Finally, adding local bag channels looping on the same process always preserves decidability, as well as adding additional bag channels in parallel to already existing ones.

Decidability. Let $\mathcal{T}$ be a topology without any jumping cycle. We apply a sequence of transformations to $\mathcal{T}$ in order to obtain a topology $\mathcal{U}$ with no FIFO channel and such that $\mathcal{T} \sqsubseteq \mathcal{U}$. Since the latter is decidable by a reduction to Petri nets, the former is decidable as well. The first transformation is to *split* all *irreversible* bag channels (cf. Sec. 3). Let $\mathcal{U}_0$ be the resulting topology.

Lemma 5.3. *The topology $\mathcal{U}_0$ does not have any jumping cycles, every FIFO channel is essential in $\mathcal{U}_0$, and $\mathcal{T} \sqsubseteq \mathcal{U}_0$.*

Let $\{c_1, \dots, c_n\}$ be the set of FIFO channels in $\mathcal{U}_0$. By the previous lemma, all c_i's are essential. Thus, they can be eliminated by fusion (cf. Section 3). For $1 \leqslant i \leqslant n$, let $\mathcal{U}_i$ be obtained from $\mathcal{U}_{i-1}$ by the fusion of c_i in $\mathcal{U}_{i-1}$.

Lemma 5.4. *For any $0 \leqslant i \leqslant n$, $\mathcal{U}_i$ does not have any jumping cycle, channels $c_{i+1}, \dots, c_n$ are essential in $\mathcal{U}_i$, and $\mathcal{U}_{i-1} \sqsubseteq \mathcal{U}_i$ when $1 \leqslant i \leqslant n$.*

We obtain the following sequence of reductions: $\mathcal{T} \sqsubseteq \mathcal{U}_0 \sqsubseteq \mathcal{U}_1 \sqsubseteq \cdots \sqsubseteq \mathcal{U}_n = \mathcal{U}$, where $\mathcal{U}$ contains only bag channels, and thus is decidable by a simple reduction to reachability in Petri nets. Thus, we have reduced reachability of communicating automata with FIFO and bag channels to reachability of Petri nets.

Undecidability. Consider a topology $\mathcal{T}$ containing a jumping cycle $p_0 \overset{c_1}{\Longleftrightarrow} q_1 \approx_D \cdots \overset{c_n}{\Longleftrightarrow} q_n \approx_D p_n = p_0$. We may assume, w.l.o.g., that $q_i \not\approx_D q_j$ for all $i \neq j$. It follows that

$$p_0 \overset{c_1}{\Longleftrightarrow} q_1 \approx_{C_1} p_1 \overset{c_2}{\Longleftrightarrow} q_2 \approx_{C_2} \cdots \overset{c_n}{\Longleftrightarrow} q_n \approx_{C_n} p_n = p_0$$

for some disjoint subsets $C_1, \dots, C_n$ of D which are supports of directed cycles in $\mathcal{T}$. We show that $\mathcal{U} \sqsubseteq \mathcal{T}$ where $\mathcal{U}$ is the cyclic topology

$$\hat{p}_0 \overset{c_1}{\Longleftrightarrow} \hat{p}_1 \overset{c_2}{\Longleftrightarrow} \cdots \overset{c_n}{\Longleftrightarrow} \hat{p}_n = \hat{p}_0$$

for which reachability is undecidable by Theorem 2.2. We build a sequence of simpler and simpler topologies $\mathcal{T}_0, \mathcal{T}_1, \dots, \mathcal{T}_n$ by fusing together synchronizable processes, as follows: $\mathcal{T}_0 = \mathcal{T}$, and $\mathcal{T}_{i+1}$ results from the fusion of C_i in $\mathcal{T}_i$, for each

$0 \leqslant i \leqslant n$. Notice that, at each step, fusing C_i in $\mathcal{T}_i$ preserves the synchronizability properties of the other processes, since $C_1, \ldots, C_n$ are disjoint. Since $\mathcal{T}_{i+1} \sqsubseteq \mathcal{T}_i$ by Proposition 4.2, we clearly have $\mathcal{U} \sqsubseteq \mathcal{T}_n \sqsubseteq \mathcal{T}_{n-1} \sqsubseteq \cdots \sqsubseteq \mathcal{T}_0 = \mathcal{T}$. We conclude that $\textsc{Reach}(\mathcal{T})$ is undecidable.

6 Extension to communicating tick automata

Communicating tick automata are communicating processes where processes can additionally synchronise over a global, rendezvous action τ [6]. In other words, when a tick automaton performs a rendezvous action τ, all other tick automata have to do the same. The additional synchronisation power provided by τ makes reachability harder to decide in general. In this section we characterise decidable topologies of tick automata over both FIFO and bag channels. This generalises our previous work [6], where only FIFO channels are considered. Let $\textsc{Reach}_\tau(\mathcal{T})$ be the reachability problem for systems of communicating tick automata with topology $\mathcal{T}$. A *strong cycle* is an undirected cycle

$$p_0 \overset{c_1}{\Longleftrightarrow} p_1 \overset{c_2}{\Longleftrightarrow} \cdots \overset{c_n}{\Longleftrightarrow} p_n = p_0$$

such that at least one channel c_i is a FIFO channel.

Theorem 6.1. *Given a topology $\mathcal{T}$, $\textsc{Reach}_\tau(\mathcal{T})$ is decidable if, and only if, $\mathcal{T}$ has no strong cycle, even when bag channels are over a unary message alphabet.*

In the next two subsections we prove both directions of the theorem.

Decidability. We give a general construction that replaces synchronisation over τ by communication over unary bag channels. In this way we reduce reachability for communicating tick automata to reachability of communicating automata (with no rendezvous action τ). This simplifies a similar construction in [6], generalizing it to the setting of FIFO and bag channels. For each channel $p \overset{c}{\Longrightarrow} q$ we add a new unary, backward bag channel denoted $q \overset{c^{-1}}{\Longrightarrow} p$. For a channel c, c^{-1} is its associated *acknowledgement channel*, and it will measure the amount of desynchronisation of q w.r.t p; the unique symbol in its alphabet will be denoted by 1 below. For a topology $\mathcal{T}$, let $\mathcal{T}^\circ$ denote the *completion* topology obtained by adding all acknowledgement channels.

Lemma 6.2. *Let $\mathcal{T}$ be a topology. Then, $\textsc{Reach}_\tau(\mathcal{T})$ is reducible to $\textsc{Reach}(\mathcal{T}^\circ)$.*

Proof. We sketch the construction here, while the full proof is in Appendix C. For a tick automaton p, let $F[p]$ denote the set of its outgoing acknowledgement channels, i.e., $F[p] = \{c^{-1} \mid q \overset{c}{\Longrightarrow} p\}$, and let $F^{-1}[p]$ denote the set of incoming acknowledgement channels, i.e., $F^{-1}[p] = \{c^{-1} \mid p \overset{c}{\Longrightarrow} q\}$. The constructions just consists in replacing a rendezvous action with the following communication over acknowledgement channels: For each tick automaton p,

$$s_0^p \xrightarrow{\tau} s_1^p \quad \text{is replaced by} \quad s_0^p \xrightarrow{F[p]!1;F^{-1}[p]?1} s_1^p$$

Here, the notation $D!1$ is a shorthand for a sequence of actions $d_1!1; d_2!1; \dots d_n!1$ if $D = \{d_1, d_2, \dots, d_n\}$, and similarly for $D?1$. Notice that in the new definition, there is no rendezvous action. This construction ensures that, for each channel $p \overset{c}{\Longrightarrow} q$, the receiver automaton q has always done at least the same number of rendezvous actions τ as the sender automaton p; if tick automata belong to the same strongly connected component, then they will even perform the same number of τ's. Moreover, since we require channels to be empty at the end of accepting runs, at the end all automata will have performed the same number of τ's.[3] This is all what is needed for preserving the causality of transmissions and receptions in the presence of τ actions, and it suffices for correctness. $\square$

Lemma 6.3. *Let $\mathcal{T}$ be a topology. $\mathcal{T}$ contains a strong cycle if, and only if, $\mathcal{T}^\circ$ contains a jumping cycle.*

Therefore, if there is no strong cycle in $\mathcal{T}$, then there is no jumping cycle in its completion $\mathcal{T}^\circ$, and by Lemma 6.2 and Theorem 5.2 we reduce to a decidable instance of the reachability problem for communicating automata (with no rendezvous action).

Undecidability. In Proposition 4.2 we have shown that processes along a directed cycle can simulate rendezvous synchronisation over two actions a and b. Here, we show that in the presence of a global rendezvous action τ, it suffices that tick automata are connected with an undirected path (even over unary bag channels).

Proposition 6.4. *Let D be the support of an undirected path in a topology $\mathcal{T}$, and let $\mathcal{U}$ be the topology resulting from the fusion of D in $\mathcal{T}$. Then,* $\textsc{Reach}_\tau(\mathcal{U})$ *is reducible to* $\textsc{Reach}_\tau(\mathcal{T})$.

This implies undecidability in the presence of a strong cycle: Let $p_0 \overset{c_1}{\Longleftrightarrow} p_1 \cdots \overset{c_n}{\Longleftrightarrow} p_0$ be a strong cycle in a topology $\mathcal{T}$. W.l.o.g., we can assume that c_1 is a FIFO channel. Then, we fuse the remaining undirected path $p_1 \overset{c_2}{\Longleftrightarrow} \cdots \overset{c_n}{\Longleftrightarrow} p_0$ to obtain an undecidable topology $\mathcal{U}$ with a FIFO self-loop $\hat{p} \overset{c_1}{\Longleftrightarrow} \hat{p}$. By Proposition 6.4, $\mathcal{T}$ is undecidable.

We now explain how to synchronise tick automata along an undirected path. Intuitively, τ is used to force rendezvous synchronisation over either a or b, and the content of the connecting channels determines whether a or b is actually performed. Let $p = p_0 \overset{c_1}{\Longleftrightarrow} p_1 \cdots \overset{c_n}{\Longleftrightarrow} p_n = q$ be an undirected path between p and q. The generic protocol for automaton p_i is shown in Fig. 5. The notation $c|m$ means $c?m$ or $c!m$ depending on whether the automaton is a sender or a receiver on channel c. For the border cases $i = 1$ or $i = n$, the corresponding actions on c_0 and c_{n+1} should be ignored.

This protocol is similar to the one from Sec. 4, but simpler thanks to the stronger synchronisation mechanism provided by the global rendezvous action τ.

[3] There is a synchronisation problem at the end of a run if $\mathcal{T}$ is not connected. This can be easily resolved by adding acknowledgement channels between the disconnected components to connect them.

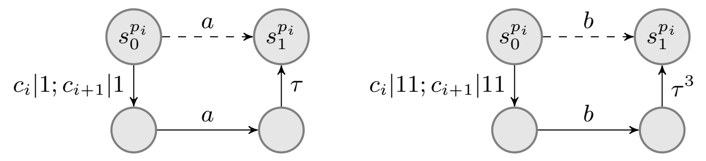


Fig. 5: Synchronisation protocol for tick automaton p_i

We now argue about its correctness. By way of contradiction, assume that the first mismatch happens between two automata $p_{i-1} \overset{c_i}{\Longrightarrow} p_i$. (The case $p_{i-1} \overset{c_i}{\Longleftarrow} p_i$ is similar.) Since the protocol was correct until then, all channels are currently empty. On the one hand, if automaton p_{i-1} simulates action a, then one message is sent on c_i and one rendezvous action τ is performed, therefore p_i cannot simulate action b which requires two messages in c_i. On the other hand, if automaton p_{i-1} simulates action b, then two messages are sent on c_i and three rendezvous actions τ are performed. In this case, assume p_i simulate action a, by performing one rendezvous action τ, and leaving one message in c_i. Since just one message is left in c_i, p_i can only simulate another action a, after which c_i is empty. However, it will have performed only two rendezvous actions τ vs. the three τ's of p_{i-1}. But c_i is now empty, and automaton p_i is not able to continue the protocol.

7 Bag channels versus lossy channels

In [5], the authors consider topologies mixing perfect and lossy[4] FIFO channels. They obtain a complete characterisation of topologies for which reachability is decidable by reducing to basic decidable topologies. Two reduction rules are introduced. The first one is the fusion of essential channels, which we used in Proposition 3.3. The second one is splitting a topology $\mathcal{T}$ into $\mathcal{T}_1$ and $\mathcal{T}_2$ when all channels between $\mathcal{T}_1$ and $\mathcal{T}_2$ are unidirectional and lossy. Despite similar names, this is unrelated to splitting irreversible channels introduced in Sec. 3. In [5], the authors split *topologies* into smaller topologies which are easier for reachability. In our case, we split *channels* to make more channels essential, which allows to fuse more processes. We note here that a lossy channel $p \overset{c}{\Longrightarrow} q$ can never be split into two lossy channels $p \overset{c_0}{\Longrightarrow} r \overset{c_1}{\Longleftarrow} q$. Therefore, our techniques are incomparable.

Reachability is decidable when all channels are lossy [1]. It is also decidable for the simple topologies in Fig. 1a and 3a when one of the two channels is lossy [5]. We obtain similar results if we replace lossy channels with bag channels.

However, some topologies which are undecidable with lossy channels become decidable with bag channels. The topology in Fig. 3a with one perfect and one lossy channel is decidable. But adding another lossy channel from p to q results in a topology which is undecidable. Similarly, adding a lossy channel looping on either p or q also makes the topology undecidable. Replacing lossy channels with

[4] Lossy FIFO channels non-deterministically lose messages.

bag channels makes all these topologies decidable. More generally, adding several unidirectional bag channels in parallel or adding bag channel loops on processes never leads to undecidability in our setting.

Finally, while the topology in Fig. 2 is undecidable when channels b_0 and b_1 are either both bag channels or both lossy channels, our construction with unary bag channels is correct *even if those are unary and lossy*. Indeed, ... However, the construction from [5] does not generalise to unary channels in this case. Thus, we strengthen their undecidability result for this topology.

REVIEW 2: add a bit of explanation on why this is the case.

8 Conclusions and future work

We have presented a complete characterisation of the decidable topologies for networks of finite-state and discrete-time automata communicating over FIFO and bag channels. Our results rely on the standard technique of fusing essential channels, and on the novel technique of splitting irreversible channels.

We have compared our work to [5], where they solve the same characterisation problem but for networks of perfect and lossy FIFO channels. An interesting direction for future research is to characterise topologies where each channel can either be perfect or lossy, and, independently, FIFO or bag.

Relaxing FIFO channels to the bag type can be applied in other contexts as well. For example, the work [8] studies topologies of networks of *pushdown automata* communicating over FIFO channels, and it is natural to ask what happens when some channel and/or pushdown stores are bags instead of strings.

References

1. P. A. Abdulla and B. Jonsson. Verifying programs with unreliable channels. *Inf. Comput.*, 127(2):91–101, 1996.
2. D. Brand and P. Zafiropulo. On communicating finite-state machines. *J. ACM*, 30(2):323–342, 1983.
3. G. Cécé and A. Finkel. Verification of programs with half-duplex communication. *Inf. Comput.*, 202(2):166–190, 2005.
4. G. Cécé, A. Finkel, and S. Purushothaman Iyer. Unreliable channels are easier to verify than perfect channels. *Inf. Comput.*, 124(1):20–31, 1996.
5. P. Chambart and Ph. Schnoebelen. Mixing lossy and perfect fifo channels. In *Proc. CONCUR*, pages 340–355, 2008.
6. L. Clemente, F. Herbreteau, A. Stainer, and G. Sutre. Reachability of communicating timed processes. In *Proc. FoSSaCS*, pages 81–96, 2013.
7. C. Haase, S. Schmitz, and Ph. Schnoebelen. The power of priority channel systems. In *CONCUR*, pages 319–333, 2013.
8. A. Heußner, J. Leroux, A. Muscholl, and G. Sutre. Reachability analysis of communicating pushdown systems. *LMCS*, 8(3):1–20, 2012.
9. S. Rao Kosaraju. Decidability of reachability in vector addition systems (preliminary version). In *Proc. STOC*, pages 267–281, 1982.
10. P. Krcál and W. Yi. Communicating timed automata: The more synchronous, the more difficult to verify. In *Proc. CAV*, pages 249–262, 2006.

11. S. La Torre, P. Madhusudan, and G. Parlato. Context-bounded analysis of concurrent queue systems. In *Proc. TACAS*, pages 299–314, 2008.
12. J. Leroux. Vector addition system reachability problem: a short self-contained proof. In *Proc. POPL*, pages 307–316, 2011.
13. M. Lohrey and A. Muscholl. Bounded MSC communication. *Inf. Comput.*, 189(2):160 – 181, 2004.
14. E. W. Mayr. An algorithm for the general petri net reachability problem. In *Proc. STOC*, pages 238–246, 1981.
15. R. Milner. *Communication and Concurrency.* Prentice Hall, 1989.
16. J. K. Pachl. Reachability problems for communicating finite state machines. Research Report CS-82-12, University of Waterloo, May 1982.
17. W. Peng and S. Purushothaman. Analysis of a class of communicating finite state machines. *Acta Informatica*, 29:499–522, 1992.
18. K. Sen and M. Viswanathan. Model checking multithreaded programs with asynchronous atomic methods. In *Proc. CAV*, pages 300–314, 2006.

A Proofs for Section 3

Lemma 3.2 ([5]). *If c is an essential channel, then every run starting and ending with c empty is order-equivalent to a run that is synchronous for c.*

Proof. By induction on the length of runs. The basis is trivial. Consider a run ρ, of non-zero length, that starts and ends with c empty. If the first action of ρ is not a transmission on c, then we obtain an order-equivalent run that is synchronous on c by applying the induction hypothesis to the remainder of ρ. Assume, on the contrary, that ρ starts with a transmission on c. The run ρ may be decomposed into $\rho = x_1 \xrightarrow{c!m} x_1' \cdot \rho_1 \cdot x_2 \xrightarrow{c?m} x_2' \cdot \rho_2$ where ρ_1 contains no reception on c.

Let p and q denote the source and target of c, respectively. We first show that the run $\chi = x_1 \xrightarrow{c!m} x_1' \cdot \rho_1$ can be reordered into a run of the form $\chi_1 \cdot y_1 \xrightarrow{c!m} y_1' \cdot \chi_2$ such that χ_1 contains no move of p and χ_2 contains no move of q. Consider the system $\hat{\mathcal{S}}$ obtained from $\mathcal{S}$ by removing the channel c and transforming all communication actions on c into internal actions (written "$c!m$" and "$c?m$"). Let $\hat{\chi}$ denote the run in $\hat{\mathcal{S}}$ obtained from χ in the obvious way. Since c is essential in $\mathcal{T}$, there is no directed path from p to q in the topology $\hat{\mathcal{T}}$ of $\hat{\mathcal{S}}$. By Lemma 3.4, the run $\hat{\chi}$ is order-equivalent to a run of the form $\hat{\chi}_1 \cdot \hat{y}_1 \xrightarrow{\text{“}c!m\text{”}} \hat{y}_1' \cdot \hat{\chi}_2$ such that $\hat{\chi}_1$ contains no move of p and $\hat{\chi}_2$ contains no move of q. Observe that this run contains no "$c?m$". So it can be mapped back to $\mathcal{S}$, and yields a run that is order-equivalent to χ and of the desired form.

The run ρ is order-equivalent to the run $\chi_1 \cdot y_1 \xrightarrow{c!m} y_1' \cdot \chi_2 \cdot x_2 \xrightarrow{c?m} x_2' \cdot \rho_2$. Since χ_2 contains no move of q, the run $\chi_2 \cdot x_2 \xrightarrow{c?m} x_2'$ can be reordered into a run of the form $y_2 \xrightarrow{c?m} y_2' \cdot \chi_3$. So the run ρ is order-equivalent to $\chi_1 \cdot y_1 \xrightarrow{c!m} y_1' \cdot y_2 \xrightarrow{c?m} y_2' \cdot \chi_3 \cdot \rho_2$. Since χ_1 contains no move of p, the channel c is empty in y_2'. It follows from the induction hypothesis that $\chi_3 \cdot \rho_2$ is order-equivalent to a run that is synchronous for c, hence, so does ρ. □

Lemma 3.4. *Given a process p, every run is order-equivalent to a run that is causal for p.*

Proof. We show, by induction on the length of runs, that every run ρ satisfies the following property: for every process p, there exists a run causal for p that is order-equivalent to ρ. This property obviously holds for runs of length at most one. Consider a run ρ of length two, written as $\rho = t_p \cdot t_q$, where t_p and t_q are moves of processes p and q, respectively. This run is obvioully causal for every process distinct from q. If $p \overset{*}{\Longrightarrow} q$, then it is also causal for q. Otherwise, $p \neq q$ and there is no channel with source p and destination q. So the moves of p and q can be swapped in ρ, yielding an order-equivalent run that is causal for q.

Consider now a run ρ of length at least three, and let p be a process. If ρ contains no move of p, then ρ is trivially causal for p. So we assume, for the remainder of the proof, that ρ contains a move of p. We first show that ρ is order-equivalent to a run whose first move is by a process r such that $r \overset{*}{\Longrightarrow} p$.

The run ρ may be written as $\rho = t \cdot \chi$, where t is a transition and χ is a run of length at least two. If χ contains no move of p, then the transition t, which is the first move of ρ, is a move of p. Otherwise, χ contains a move of p. According to the induction hypothesis, χ is order-equivalent to a run χ' that is causal for p. The run χ', whose length is at least two, may be written as $\chi' = t_q \cdot \chi'' \cdot t'$ where t_q is a move of a process q such that $q \overset{*}{\Longrightarrow} p$. Notice that ρ is order-equivalent to $t \cdot t_q \cdot \chi'' \cdot t'$. By applying the induction hypothesis to the process q and the run $t \cdot t_q \cdot \chi''$, we get a run ρ' causal for q that is order-equivalent to $t \cdot t_q \cdot \chi''$. Observe that ρ is order-equivalent to $\rho' \cdot t'$. Moreover, since ρ' causal for q and contains a move of q, the first move of ρ' is by a process r such that $r \overset{*}{\Longrightarrow} q \overset{*}{\Longrightarrow} p$. We have shown that ρ is order-equivalent to a run $t_r \cdot \rho''$ whose first move t_r is by a process r such that $r \overset{*}{\Longrightarrow} p$. Replacing ρ'' by an order-equivalent run that is causal for p (which is possible by the induction hypothesis) concludes the proof of the lemma. □

Proposition 3.6 (Split). *If c is an irreversible channel in a topology $\mathcal{T}$, then $\mathcal{T} \sqsubseteq \mathcal{U}$ where $\mathcal{U}$ results from the split of c in $\mathcal{T}$.*

Proof. Let $\mathcal{S}$ be a system with topology $\mathcal{T}$. We construct a new system $\mathcal{S}'$ with topology $\mathcal{U}$ as follows. The new process r will just match messages from c_0 and c_1: It has state s_0 which is both initial and final, and intermediate states $\{s_m\}_{m \in M}$ for each message $m \in M$, along with transitions $s_0 \xrightarrow{c_0?m} s_m \xrightarrow{c_1?m} s_0$. The definition of p and q in $\mathcal{S}'$ is the same as in $\mathcal{S}$, except that 1) transmissions $c!m$ of p are replaced by transmissions $c_0!m$, and 2) receptions $c?m$ of q are replaced by transmissions $c_1!m$. It is immediate from the definition that an accepting run in $\mathcal{S}$ can be step-wise translated into a run in $\mathcal{S}'$ that ends with the same contents in c_0 and c_1. This latter run is extended to an accepting run with the additional actions of r to match messages in c_0 and c_1.

On the other side, an accepting run in $\mathcal{S}'$ induces an accepting run in $\mathcal{S}$ by reordering. Let π' be an accepting run in $\mathcal{S}'$. By Lemma 3.4, π' can be reordered into an accepting run π'' in $\mathcal{S}'$ which is causal for p. Since there is no directed path from q to p and π'' is causal for p, each move of p occurs before each move of q. That is, in π'' all transmissions on c_0 occur before any transmission on c_1 (causality), and the definition of r ensures that the received string equals the sent string. Therefore, we massage π'' as follows: 1) replace transmissions $c_0!m$ of p with $c!m$, 2) replace transmissions $c_1!m$ of q with receptions $c?m$, and 3) replace the actions of r, and we get an accepting run π in $\mathcal{S}$. □

B Proofs for Section 5

We begin with an observation which does not appear as a lemma in the main text. We show that adding acknowledgement channels for each essential FIFO channel preserves decidability.

Lemma B.1. *For a topology $\mathcal{T}$, let $\mathcal{U}$ be the topology is obtained from $\mathcal{T}$ by adding an acknowledgement channel for each essential FIFO channel of $\mathcal{T}$. Then, $\mathcal{T}$ has a jumping cycle iff $\mathcal{U}$ has a jumping cycle.*

Proof. Clearly, if $\mathcal{T}$ has a jumping cycle, so it does $\mathcal{U}$, as the latter contains more channels. Now, let $\mathcal{U}$ have a jumping cycle

$$p_0 \overset{c_1}{\Longleftrightarrow} q_1 \approx_D p_1 \overset{c_2}{\Longleftrightarrow} q_2 \approx_D \cdots \overset{c_n}{\Longleftrightarrow} q_n \approx_D p_n = p_0 \ .$$

We show that, by removing one acknowledgement channel, we still have a jumping cycle. The result will follow by induction on the number of acknowledgement channels removed from D. Let $p \overset{c}{\Longrightarrow} q$ be the essential FIFO channel for which the acknowledgement channel $q \overset{c^{-1}}{\Longrightarrow} p$ was added. Since c^{-1} is a bag channel, it is different from all c_i's. However, c^{-1} can be in D. Let $E = D \backslash \{c^{-1}\}$. Consider any pair of synchronizable processes $q_i \approx_D p_i$ in $\mathcal{U}$. If $q_i \approx_E p_i$ holds in $\mathcal{T}$, we are done. Otherwise, let $q_i \not\approx_E p_i$ in $\mathcal{T}$. In this case, we show that

$$q_i \approx_E q \overset{c}{\Longleftarrow} p \approx_E p_i$$

holds in $\mathcal{T}$, thus giving rise to a jumping cycle therein. Indeed, since $q_i \approx_D p_i$ in $\mathcal{U}$, there exists a simple directed path from q_i to p_i, and a simple directed path from p_i to q_i. At least one of those two paths must contain c^{-1} (since $q_i \not\approx_E p_i$): W.l.o.g., say the simple directed path from q_i to p_i contains $q \overset{c^{-1}}{\Longrightarrow} p$. This implies

$$q_i \overset{*}{\Longrightarrow} q \overset{c^{-1}}{\Longrightarrow} p \overset{*}{\Longrightarrow} p_i$$

where the two directed paths from q_i to q and from p to p_i do not contain c^{-1} (or c) by simplicity. Since $q_i \approx_E p_i$, there exists a simple directed path $p \overset{*}{\Longrightarrow} p_i \overset{*}{\Longrightarrow} q_i \overset{*}{\Longrightarrow} q$ in $\mathcal{U}$ from p to q. If the simple directed path $p_i \overset{*}{\Longrightarrow} q_i$ contains the acknowledgement channel $q \overset{c^{-1}}{\Longrightarrow} p$, i.e., $p_i \overset{*}{\Longrightarrow} q \overset{c^{-1}}{\Longrightarrow} p \overset{*}{\Longrightarrow} q_i$, then there exists a directed path from p to q not containing neither c^{-1} nor c, contradicting the essentiality of c. Otherwise, the simple directed path $p_i \overset{*}{\Longrightarrow} q_i$ does not contain the acknowledgement channel $q \overset{c^{-1}}{\Longrightarrow} p$: In this case, there exists a directed path from p to q in $\mathcal{T}$. Since $p \overset{c}{\Longrightarrow} q$ is essential, this path must contain c. I.e.,

$$p_i \overset{*}{\Longrightarrow} p \overset{c}{\Longrightarrow} q \overset{*}{\Longrightarrow} q_i \text{ in } \mathcal{T}$$

Thus, $q_i \approx_E q$ and $p \approx_E p_i$. □

Lemma 5.3. *The topology $\mathcal{U}_0$ does not have any jumping cycles, every FIFO channel is essential in $\mathcal{U}_0$, and $\mathcal{T} \sqsubseteq \mathcal{U}_0$.*

Notice that in $\mathcal{U}_0$ all irreversible bag channels are of the form $p \Longrightarrow r$, with r an end-component in the topology.

Proof. While splitting channels in general eliminates directed paths, it does not change the synchronizability equivalence[5] $\approx$ as only channels between non-synchronizable processes are split. As a consequence, splitting irreversible channels cannot introduce jumping cycles in $\mathcal{U}_0$.

By a repeated application of Proposition 3.6 to each irreversible bag channel which is split, we have $\mathcal{T} \sqsubseteq \mathcal{U}_0$.

Finally, we show that all FIFO channels are essential in $\mathcal{U}_0$. Indeed, consider a FIFO channel $p \overset{c}{\Longrightarrow} q$. By way of contradiction, assume that c is not essential. By definition, there exists a directed path from p to q not containing c. Only two cases can occur.

– Let $p \approx q$. There exists a simple directed path from q to p (necessarily not containing c). This implies $p \approx_C q$ with $c \notin C$, and one has a jumping cycle $p \overset{c}{\Longrightarrow} q \approx_C p$ in $\mathcal{U}_0$, which is a contradiction.
– Let $p \not\approx q$. Since p and q are not synchronizable, the directed path from p to q needs to take a channel $r \overset{d}{\Longrightarrow} s$ with $r \not\approx s$. Let $r_1 \overset{d_1}{\Longrightarrow} s_1, \dots, r_k \overset{d_k}{\Longrightarrow} s_k$ be all those channels which jump over different equivalence classes $r_i \not\approx s_i$ in the directed path from p to q. We clearly have the following arrangement:

$$p \approx r_1 \overset{d_1}{\Longrightarrow} s_1 \approx r_2 \overset{d_2}{\Longrightarrow} s_2 \cdots r_k \overset{d_k}{\Longrightarrow} s_k \approx q \overset{c}{\Longleftarrow} p \ .$$

If all d_i's are FIFO, then we immediately have a jumping cycle, which is a contradiction. Thus, one d_i is an irreversible bag channel between non-synchronizable processes $r_i \not\approx s_i$. By the definition of $\mathcal{U}_0$, the process s_i is an end-component in the topology, which contradicts the existence of a directed path from s_i to r_{i+1} (or q if $i = k$). □

Lemma 5.4. *For any $0 \leqslant i \leqslant n$, $\mathcal{U}_i$ does not have any jumping cycle, channels $c_{i+1}, \dots, c_n$ are essential in $\mathcal{U}_i$, and $\mathcal{U}_{i-1} \sqsubseteq \mathcal{U}_i$ when $1 \leqslant i \leqslant n$.*

Proof. We proceed by induction on i. The base case $i = 0$ holds by Lemma 5.3. For the inductive step, assume the statement holds for $0 \leqslant i-1 \leqslant n-1$, and we show it for i. Since c_i is essential in $\mathcal{U}_{i-1}$ by induction hypothesis, $\mathcal{U}_{i-1} \sqsubseteq \mathcal{U}_i$ holds by Proposition 3.3.

We prove that $\mathcal{U}_i$ does not have any jumping cycle. By way of contradiction, assume that $\mathcal{U}_i$ has a jumping cycle

$$r_0 \overset{d_1}{\Longleftrightarrow} s_1 \approx_D r_1 \overset{d_2}{\Longleftrightarrow} s_2 \approx_D \cdots \overset{d_n}{\Longleftrightarrow} s_n \approx_D r_n = r_0 \ .$$

We show that it induces a jumping cycle in $\mathcal{U}_{i-1}$. When going from $\mathcal{U}_{i-1}$ to $\mathcal{U}_i$ we have fused the essential FIFO channel $p \overset{c_i}{\Longrightarrow} q$. Seen the other way around, when going from $\mathcal{U}_i$ to $\mathcal{U}_{i-1}$ we take a process t in the former, we partition its incoming and outgoing channels into two sets D_0 and D_1, and we replace t by $p \overset{c_i}{\Longleftrightarrow} q$, letting p take the channels in D_0 and q the channels in D_1. We show that if $r \approx_D s$ in $\mathcal{U}_i$, then either $r \approx_D s$ or $r \approx_{D_0} p \overset{c_i}{\Longleftrightarrow} q \approx_{D_1} s$ in $\mathcal{U}_{i-1}$.

[5] Except for adding a singleton equivalence class for the new process $[r] = \{r\}$.

Assume $r \approx_D s$ in $\mathcal{U}_i$. If $r \approx_D s$ in $\mathcal{U}_{i-1}$ we are done.

Otherwise, let $r \not\approx_D s$ in $\mathcal{U}_{i-1}$. W.l.o.g., assume there is no simple directed path from s to r in $\mathcal{U}_{i-1}$. However, there is such a simple directed path in $\mathcal{U}_i$, which it has to pass through t:

$$\pi_0 = s \xRightarrow{e_1} s_1 \xRightarrow{e_2} \cdots t \cdots s_{k-1} \xRightarrow{e_k} r \text{ in } \mathcal{U}_i$$

Therefore, we get the following simple undirected path in $\mathcal{U}_{i-1}$, where t is expanded into $q \xLeftarrow{c_i} p$:

$$\pi_1 = s \xRightarrow{e_1} s_1 \xRightarrow{e_2} \cdots q \xLeftarrow{c_i} p \cdots s_{k-1} \xRightarrow{e_k} r \text{ in } \mathcal{U}_{i-1}$$

(Process t cannot be expanded into $p \xRightarrow{c_i} q$ since by assumption there is no directed path from s to r in $\mathcal{U}_{i-1}$.) By simplicity, channel c does not appear anywhere else in the path above. Thus, there exist directed paths from s to q, and from p to r in $\mathcal{U}_{i-1}$:

$$s \xRightarrow{*} q \quad \text{and} \quad p \xRightarrow{*} r \quad \text{in } \mathcal{U}_{i-1}$$

On the one hand, assume there exists a simple directed path from r to s in $\mathcal{U}_{i-1}$. Then, there exists a path from p to q in $\mathcal{U}_{i-1}$. But c_i is essential. Thus, the latter path has to pass through c_i. Since the former undirected path from s to r is simple, c_i has to appear in the directed path from r to s. Thus, $\mathcal{U}_{i-1}$ contains the following simple directed path:

$$r \xRightarrow{f_1} r_1 \xRightarrow{f_2} \cdots p \xRightarrow{c_i} q \cdots r_{h-1} \xRightarrow{f_h} s \text{ in } \mathcal{U}_{i-1}$$

By simplicity, channel c_i does not appear anywhere else in the path above. Therefore, there are directed paths from r to p and from q to s in $\mathcal{U}_{i-1}$ not containing c_i, showing $r \approx_{D_0} p \xLeftrightarrow{c_i} q \approx_{D_1} s$ in $\mathcal{U}_{i-1}$.

On the other hand, assume there is no directed path from r to s in $\mathcal{U}_{i-1}$. We show that this case cannot occur. Since there is a directed path from r to s in $\mathcal{U}_i$, we get the following simple undirected path in $\mathcal{U}_{i-1}$ (where c_i appears in the "wrong" direction):

$$r \xRightarrow{f_1} r_1 \xRightarrow{f_2} \cdots q \xLeftarrow{c_i} p \cdots r_{l-1} \xRightarrow{f_l} s \text{ in } \mathcal{U}_{i-1}$$

By combining the directed path $p \xRightarrow{*} s$ above with the directed path $s \xRightarrow{*} q$ from π_1 above, we get a directed path from p to q not containing c_i, which contradicts the essentiality of the latter. □

C Proofs of Section 6

Lemma 6.2. *Let $\mathcal{T}$ be a topology. Then,* $\text{REACH}_\tau(\mathcal{T})$ *is reducible to* $\text{REACH}(\mathcal{T}^\circ)$.

Proof. Let $\mathcal{S}$ be a system of communicating tick automata with topology $\mathcal{T}$, and let $\mathcal{S}'$ be the system of communicating automata obtained by adding acknowledgement channels, and replacing transitions $s_0^p \xrightarrow{\tau} s_1^p$ with

$$s_0^p \xrightarrow{F[p]!1} t_{s_1}^p \xrightarrow{F^{-1}[p]?1} s_1^p$$

where $t_{s_1}^p$ is a new, non-accepting intermediate state that depends on s_1^p.

We view a run in $\mathcal{S}$ (or in $\mathcal{S}'$) as a sequence of the local transitions as performed by each process, i.e., a run is a sequence $\pi = t_0, t_1, \ldots, t_n$, where each transition t_i is fired by some process in P. For each process p, let $\pi|_p$ be the projection of π containing only transitions fired by process p.

Any accepting run π in $\mathcal{S}$ can be transformed into an accepting run π' in $\mathcal{S}'$. Rendezvous actions τ are synchronised in π, i.e., they always occur in blocks of length n:

$$\pi = \rho_1 \xi_1 \rho_2 \xi_2 \cdots \rho_n$$

where no τ action appears in the ρ_i's, and each ξ_j is of the form

$$\xi_j = s_0^{p_1} \xrightarrow{\tau} s_1^{p_1}, s_0^{p_2} \xrightarrow{\tau} s_1^{p_2}, \ldots, s_0^{p_k} \xrightarrow{\tau} s_1^{p_k}$$

where each process p has exactly one τ transition (assuming there are k processes in P). Rendezvous actions τ are replaced by transmissions of all acknowledgement messages, followed by their matching receptions. If we let ξ_j' be defined as

$$xi_j' = s_0^{p_1} \xrightarrow{F[p_1]!1} t_{s_1}^{p_1}, \ldots, s_0^{p_k} \xrightarrow{F[p_k]!1} t_{s_1}^{p_k}, t_{s_1}^{p_1} \xrightarrow{F^{-1}[p_1]?1} s_1^{p_1}, \ldots, t_{s_1}^{p_k} \xrightarrow{F^{-1}[p_k]?1} s_1^{p_k} ,$$

then we obtain π' just as

$$\pi' = \rho_1 \xi_1' \rho_2 \xi_2' \cdots \rho_n .$$

Notice that π' is a valid run in $\mathcal{S}'$ since all receptions on acknowledgement channels are always preceded by their matching transmissions. Moreover, just before and just after of blocks ξ_j', acknowledgement channels are empty. Therefore, if π is accepting, then so it is π'.

Any accepting run π' in $\mathcal{S}'$ can be transformed into an accepting run π in $\mathcal{S}$. Let π' be the sequence of transitions $\pi' = t_0, t_1, \ldots, t_n$, and let each transition t_i be of the form

$$t_i = s_i^{p_i} \xrightarrow{b_i} s_{i+1}^{p_i} .$$

The idea is to decorate transitions t_i in π' with an integral *timestamp* $k_i(p) \geqslant 0$ counting how many τ's have been simulated so far by process p. Formally, $k_i(p)$ is the number of transitions t_j in $\pi'|_p$ s.t. $j < i$ and $b_j = F[p]!1$. A few observations are in order:

- At the beginning, $k_0(p) = 0$ for every process p.
- At the end, $k_n(p) = k_n(q)$ for every process p and q (since all channels are empty at the end, and in particular acknowledgement channels).

– Timestamps are *locally non-decreasing.* I.e., for each process p,

$$k_0(p) \leqslant k_1(p) \leqslant \cdots \leqslant k_n(p) \ .$$

– For every channel $p \overset{c}{\Longrightarrow} q$ in the topology $\mathcal{T}$,

$$k_i(p) \leqslant k_i(q) \ .$$

This holds because when a process p wants to simulate a rendezvous action it has to eventually execute $F^{-1}[p]?1$. But $c \in F^{-1}$, and also q has to simulate a rendezvous action.

However, while timestamps are locally non-decreasing, they are not necessarily *globally non-decreasing*, i.e., $k_i(p_i) \leqslant k_j(p_j)$ for every $i \leqslant j$. Having globally non-decreasing timestamps is necessary to show that the processes can be correctly synchronised on τ's. We produce another run π'' starting from π', where timestamps are not only locally non-decreasing, but also globally non-decreasing. To do so, we show that transitions in π' can be swapped when the timestamp decreases. Formally, we say that a pair of adjacent transitions t_i, t_{i+1} is *offending* iff

$$k_i(p_i) > k_{i+1}(p_{i+1}) \ .$$

Notice that offending transitions cannot belong to the same process (since timestamps are locally non-decreasing), and cannot be a transmission $t_i = c!m$ and its matching reception $t_{i+1} = c?m$ for a channel $p \overset{c}{\Longrightarrow} q$ since $k_i(p_i) \leqslant k_i(p_{i+1}) = k_{i+1}(p_{i+1})$ by the observation above. Thus, all offending pairs can be swapped, and the swapping process terminates since the total number of offending pairs decreases at each step. Clearly, when no more transitions can be swapped, we have globally non-decreasing timestamps.

In a path with no offending transitions, once a process p executes an action $F[p]!1$, by simulating a rendezvous action, it is blocked until all other processes have done the same. Thus, we can replace each transition $s_0^p \xrightarrow{F[p]!1} t_{s_1}^p$ in π'' with a corresponding rendezvous action $s_0^p \xrightarrow{\tau} s_1^p$ (and remove all transitions $t_{s_1}^p \xrightarrow{F^{-1}[p]?1} s_1^p$), and obtain a sequence π which is a run in $\mathcal{S}$ where automata properly synchronise over the rendezvous action τ. Moreover, if π' was accepting, then so it is π. $\square$

Lemma 6.3. *Let $\mathcal{T}$ be a topology. $\mathcal{T}$ contains a strong cycle if, and only if, $\mathcal{T}^\circ$ contains a jumping cycle.*

Proof. On the one hand, if $\mathcal{T}$ contains a strong cycle

$$p_0 \overset{c_1}{\Longleftrightarrow} p_1 \cdots \overset{c_n}{\Longleftrightarrow} p_n = p_0$$

then by adding acknowledgement channels c_i^{-1}'s we obtain $p_1 \approx p_n$ in $\mathcal{T}^\circ$, thus forming a jumping cycle in the latter topology. On the other hand, if $\mathcal{T}^\circ$ contains a jumping cycle

$$p_0 \overset{c_0}{\Longleftrightarrow} q_0 \approx p_1 \overset{c_1}{\Longleftrightarrow} q_1 \approx \cdots \approx p_n \overset{c_n}{\Longleftrightarrow} q_n = p_0$$

then by removing all acknowledgement channels from it we obtain a strong cycle in $\mathcal{T}$, since the c_i's are FIFO (and thus not acknowledgement) channels, and $q_i \approx p_{i+1}$ with acknowledgement channels implies that there exists an undirected path from q_i to p_{i+1} with no acknowledgement channel. $\square$