Guessing the Buffer Bound for k-Synchronizability
Cinzia Di Giusto, Laetitia Laversa, Etienne Lozes

To cite this version:
Cinzia Di Giusto, Laetitia Laversa, Etienne Lozes. Guessing the Buffer Bound for k-Synchronizability. International Journal of Foundations of Computer Science, 2022, 34 (8), pp.1051-1076. 10.1142/S0129054122430018. hal-03890605

HAL Id: hal-03890605
https://hal.science/hal-03890605v1
Submitted on 7 Mar 2024

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
Guessing the buffer bound for k-synchronizability

Cinzia Di Giusto¹, Laetitia Laversa¹, and Etienne Lozes¹

Université Côte d’Azur, CNRS, I3S, Sophia Antipolis, France

Abstract. A communicating system is k-synchronizable if all of the message sequence charts representing the executions can be divided into slices of k sends followed by k receptions. It was previously shown that, for a fixed given k, one could decide whether a communicating system is k-synchronizable. This result is interesting because the reachability problem can be solved for k-synchronizable systems. However, the decision procedure assumes that the bound k is fixed. In this paper we improve this result and show that it is possible to decide if such a bound k exists.

Keywords: communicating automata · MSC · synchronizability

1 Introduction

Communicating finite state machines [4] model distributed systems where participants exchange messages via FIFO buffers. Due to the unboundedness of the buffers, the model is Turing powerful as soon as there are two participants and two queues. In order to recover decidability, several works introduced restrictions on the model, for instance: lossiness of the channels [1], specific topologies, or bounded context switching [13]. Another line of research focused on analyzing the system under the assumption that the semantics is synchronous [2,8,14,6,5,12,11] or that buffers are bounded. This assumption is not as restrictive as it may seem at first, because several systems enjoy the property that their execution, although not necessarily bounded, can be simulated by a causally equivalent bounded execution. Existentially k-bounded communicating systems [10] are precisely the systems whose message sequence charts can be generated by k-bounded executions. In particular, the reachability problem is decidable for existentially k-bounded communicating systems. A limitation of this framework is that the bound k on the buffer size must be fixed. A natural question is whether the existence of such a bound can be decided. Genest, Kuske and Muscholl answered this question negatively [10]. Bouajjani et al. [3] introduced a variant of existentially k-bounded communicating systems they called k-synchronizable systems. A system is k-synchronizable if each of its execution is causally equivalent to a sequence of communication rounds composed of at most k sends followed by at most k receptions. In particular, each execution of a k-synchronizable system is causally equivalent to a k-bounded execution (provided

¹ The results in [3] have then been refined in [7]
all messages are eventually received). Like for existentially bounded systems, the 
reachability problem becomes decidable for \( k \)-synchronizable systems, and the 
membership problem - whether a given system is \( k \)-synchronizable for a fixed 
given \( k \) is decidable as well. Bouajjani et al. conjectured that the existence of a 
bound \( k \) on the size of the communication rounds was undecidable.

Instead, in this paper, we show that this problem is decidable. This result 
contrasts with the negative result about the same question for existentially bounded 
communicating systems. There is an important difference between existentially 
bounded and \( k \)-synchronizable ones that explains this situation. Existentially 
bounded systems deal with peer-to-peer communications, with one buffer per 
pair of machines, whereas \( k \)-synchronizable systems deal with mailbox communica-
ctions where one buffer per machine merges all incoming messages.

The paper is organized as follows: in the next section, we introduce prelimi-
nary definitions on communicating automata and \( k \)-synchronizable systems. In 
Section 3 we explain the general strategy for computing the bound \( k \), which is 
to compute the automata of two regular languages: the language of reachable 
exchanges, and the language of prime exchanges. In Section 4 we focus on rea-
chable exchanges, and in Section 5 on prime exchanges. Section 6 lastly computes 
the bound \( k \). Finally Section 7 concludes with some final remarks. An appendix 
with additional material and proofs is added for the reviewer convenience.

2 Preliminaries

Let \( \mathbb{V} \) be a finite set of messages and \( \mathbb{P} \) a finite set of processes exchanging 
messages. A send action, denoted \( \text{send}(p, q, v) \), designates the sending of mes-
sage \( v \) from process \( p \) to process \( q \), storing it in the queue of \( q \). Similarly, 
a receive action \( \text{rec}(q, v) \) expresses that process \( q \) pops message \( v \) from its 
queue of incoming messages. We write \( a \) to denote a send or receive action.

Let \( S = \{ \text{send}(p, q, v) \mid p, q \in \mathbb{P}, v \in \mathbb{V} \} \) be the set of send actions and 
\( R = \{ \text{rec}(q, v) \mid q \in \mathbb{P}, v \in \mathbb{V} \} \) the set of receive actions. \( S_p \) and \( R_p \) stand for 
the set of sends and receives of process \( p \) respectively.

A system is a tuple \( \mathcal{G} = ((L_p, \delta_p, l^0_p) \mid p \in \mathbb{P}) \) where, for each process \( p \), \( L_p \) is 
a finite set of local control states, \( \delta_p \subseteq (L_p \times (S_p \cup R_p) \times L_p) \) is the transition 
relation and \( l^0_p \) is the initial state. In the rest of the paper, when talking about 
a system \( \mathcal{G} \), we may also identify it with the global automaton obtained as the 
product of the process automata and denoted \( (L_\mathcal{G}, \delta_\mathcal{G}, l_0) \) where \( L_\mathcal{G} = \Pi_{p \in \mathbb{P}} L_p \) is 
the set of global control states, \( l_0 = (l^0_p)_{p \in \mathbb{P}} \) is the initial global control state and 
\( ((l_1, \cdots, l_q, \cdots, l_n), a, (l_1, \cdots, l'_q, \cdots, l_n)) \in \delta_\mathcal{G} \) iff \( (l_q, a, l'_q) \in \delta_q \) for \( q \in \mathbb{P} \).

We write \( l_q \xrightarrow{a} l'_q \) (resp. \( 1 \xrightarrow{a} 1' \)) for \( (l_q, a, l'_q) \in \delta_q \) (resp. \( (1, a, 1') \in \delta_\mathcal{G} \)). We write 
\( \frac{a_1 \cdots a_n}{a_1 \cdots a_n} \) for \( a_1 \cdots a_n \).

A configuration is a pair \((l, \text{Buf})\) where \( l = (l_p)_{p \in \mathbb{P}} \in L_\mathcal{G} \) is a global control 
state of \( \mathcal{G} \), and \( \text{Buf} = (b_p)_{p \in \mathbb{P}} \in (\mathbb{V})^\mathbb{P} \) is a vector of buffers, each \( b_p \) being a word 
over \( \mathbb{V} \). \( \text{Buf}_p \) stands for the vector of empty buffers. The mailbox semantics of a

system is defined by the two rules below.
Definition 2 (Concatenation of MSCs). Let \( \mu \) be a set of executions of a system \( S \). To stress the causal dependencies between messages we use message sequence charts (MSCs).

Definition 1 (Message Sequence Chart). A message sequence chart \( \mu \) is a tuple \( (Ev, \lambda, \prec_{po}, \prec_{src}) \) such that:

1. \( Ev \) is a finite set of events partially ordered under \( \prec_{po} \cup \prec_{src} \),
2. \( \lambda : Ev \to S \cup R \) tags each event with an action,
3. for each process \( p \), \( \prec_{po} \) induces a total order on the events of \( p \), i.e. on \( \lambda^{-1}(S_{p} \cup R_{p}) \),
4. \( (Ev, \prec_{src}) \) is the graph of a bijection between a subset of \( \lambda^{-1}(S) \) and the whole of \( \lambda^{-1}(R) \),
5. for all \( s \prec_{src} r \), there are \( p, q, v \) such that \( \lambda(s) = send(p, q, v) \) and \( \lambda(r) = rec(q, v) \).

Definition 2 (Concatenation of MSCs). Let \( \mu_1 = (Ev_1, \lambda_1, \prec_{po}^1, \prec_{src}^1) \) and \( \mu_2 = (Ev_2, \lambda_2, \prec_{po}^2, \prec_{src}^2) \) be two MSCs. Their concatenation \( \mu_1 \cdot \mu_2 \) is the MSC \( \mu = (Ev, \lambda, \prec_{po}, \prec_{src}) \) such that:

\[
\begin{align*}
& Ev = Ev_1 \cup Ev_2 \\
& \lambda = \lambda_1 \cup \lambda_2 \\
& \prec_{po} = \prec_{po}^1 \cup \prec_{po}^2 \cup \mathcal{P}\{ (e_1, e_2) \mid e_1 \in \lambda_1^{-1}(S_{p} \cup R_{p}), e_2 \in \lambda_2^{-1}(S_{p} \cup R_{p}) \} \\
& \prec_{src} = \prec_{src}^1 \cup \prec_{src}^2
\end{align*}
\]

In a sequence of actions \( e = a_1 \cdots a_n \), a send action \( a_i = send(p, q, v) \) is matched by a reception \( a_j = rec(q', v') \) (denoted by \( a_i \leadsto a_j \) if \( i < j \), \( p = p' \), \( q = q' \), \( v = v' \)), and there is \( \ell \geq 1 \) such that \( a_i \) and \( a_j \) are the \( \ell \)th actions of \( e \) with these properties respectively. A send action \( a_i \) is \textit{unmatched} if there is no matching reception in \( e \).

The MSC associated with the execution \( e = a_1 \cdots a_n \) is \( (Ev, \lambda, \prec_{po}, \prec_{src}) \) where \( Ev = \{ 1, \cdots, n \} \), \( \lambda(i) = a_i \), \( i \prec_{po} j \) iff \( i < j \) and \( \{ a_i, a_j \} \subseteq S_{p} \cup R_{p} \) for some \( p \), and \( i \prec_{src} j \) if \( a_i \leadsto a_j \).

When \( v \) is either an unmatched \( send(p, q, v) \) or a pair of matched actions \( \{ send(p, q, v), rec(q, v) \} \), we write \( proc_{p}(v) \) for \( p \) and \( proc_{R}(v) \) for \( q \). Note that \( proc_{R}(v) \) is defined even if \( v \) is unmatched. An MSC is depicted with vertical timelines (one for each process) where time goes from top to bottom. Points on the lines represent events of this process. We draw an arc between two matched events and a dashed arc to depict an unmatched send. The concatenation \( \mu_1 \cdot \mu_2 \) of two MSCs is the union of the two MSCs where, for each \( p \), all \( p \)-events of
\( \mu_1 \) are considered \( \prec_{po} \) smaller than all \( p \)-events of \( \mu_2 \). We write \( msc(e) \) for the MSC associated with the execution \( e \), and we say that a sequence of actions \( e \) is a linearization of a given MSC if it is the sequence of actions induced by a total order extending \(( \prec_{po} \cup \prec_{src})^* \). We write \( asTr(\mathcal{S}) \) for the set \{ \( msc(e) \mid e \in asEx(\mathcal{S}) \) \}. We write \( l_{\mu} \rightarrow l'_{\mu} \) to denote that \( l_{\mu} = l_{\mu'} \Rightarrow l'_{\mu} \) for any linearization \( e \) of \( \mu \). Finally, we recall from [7] the definition of causal delivery that allows to consider only MSCs that correspond to executions in the mailbox semantics.

**Definition 3 (Causal delivery).** Let \( \mu = (Ev, \lambda, \prec_{po}, \prec_{src}) \) be an MSC. We say that \( \mu \) satisfies causal delivery if it admits a linearization with total order \(<\) such that for any two events \( s_1, s_2 \in Ev \), if \( s_1 < s_2 \), \( \lambda(s_1) = \text{send}(p, q, v) \) and \( \lambda(s_2) = \text{send}(p', q, v') \) for a same destination process \( q \), then either \( s_2 \) is unmatched, or there are \( r_1, r_2 \) such that \( s_1 \prec_{src} r_1, s_2 \prec_{src} r_2 \), and \( r_1 < r_2 \).

A \( k \)-exchange (with \( k \geq 1 \)) is an MSC that admits a linearization \( e \in S^{\leq k}R^{\leq k} \) starting with at most \( k \) sends and followed by at most \( k \) receives. An MSC is \( k \)-synchronous if it can be chopped into a sequence of \( k \)-exchanges.

**Definition 4 (k-synchronous).** An MSC \( \mu \) is \( k \)-synchronous if \( \mu = \mu_1 \cdot \mu_2 \cdot \cdots \cdot \mu_n \) where, for all \( i \in [1..n] \), \( \mu_i \) is a \( k \)-exchange.

For instance, the MSC \( \mu_1 \) depicted on Fig. 1 is 2-synchronous, as it can be split in two 2-exchanges.

An execution \( e \) is \( k \)-synchronizable if \( msc(e) \) is \( k \)-synchronous. A system \( \mathcal{S} \) is \( k \)-synchronous if all its executions are \( k \)-synchronizable.

**Theorem 1 ([3,7]).** It is decidable whether a system \( \mathcal{S} \) is \( k \)-synchronizable for a given \( k \). Moreover, it is decidable to know whether a control state is reachable under the assumption that \( \mathcal{S} \) is \( k \)-synchronizable.

This result is interesting but somehow incomplete as it assumes that a fixed value of the parameter \( k \) has been found. We aim at answering this limitation by computing the synchronizability degree of a given system.

**Definition 5 (Synchronizability degree).** The synchronizability degree \( sd(\mathcal{S}) \) of a system \( \mathcal{S} \) is the smallest \( k \) such that \( \mathcal{S} \) is \( k \)-synchronizable. In particular, \( sd(\mathcal{S}) = \infty \) if \( \mathcal{S} \) is not \( k \)-synchronizable for any \( k \).

### 3 Largest prime reachable exchange

In this section, we relate the synchronizability degree of a system to the size of a “maximal, prime, reachable exchange”. We start with defining these notions.

An exchange is a \( k \)-exchange for some arbitrary \( k \), and we call \( k \) the size of the exchange. An exchange \( \mu \) is reachable if there exist exchanges \( \mu_1, \cdots, \mu_n \) for
some \( n \geq 0 \) and such that \( \mu_1 \cdots \mu_n \cdot \mu \in asTr(\mathcal{S}) \). An exchange \( \mu \) is prime if there does not exist a decomposition \( \mu = \mu_1 \cdot \mu_2 \) in two non-empty exchanges. For instance, the 2-exchange (depicted by the MSC \( \mu_2 \), Fig. 2) with linearization:

\[
\text{send}(p, q, v_1) \cdot \text{send}(r, q, v_2) \cdot \text{rec}(q, v_1) \cdot \text{rec}(q, v_2)
\]

is not prime, as it can be factored in two 1-exchanges as follows

\[
\text{send}(p, q, v_1) \cdot \text{rec}(q, v_1) \cdot \text{send}(r, q, v_2) \cdot \text{rec}(q, v_2).
\]

The size of the biggest prime reachable exchange is related to the synchronizability degree \( sd(\mathcal{S}) \) by the following property.

**Lemma 1.** Let \( k \in \mathbb{N} \cup \{\infty\} \) be the supremum of the sizes of all prime reachable exchanges. (1) If \( k = \infty \), then \( sd(\mathcal{S}) = \infty \) (2) if \( k < \infty \), then either \( \mathcal{S} \) is \( k \)-synchronizable and \( sd(\mathcal{S}) = k \), or \( \mathcal{S} \) is not \( k \)-synchronizable and \( sd(\mathcal{S}) = \infty \).

**Proof.** Let \( k \in \mathbb{N} \cup \{\infty\} \) be the supremum of the sizes of all prime reachable exchanges.

Assume that there exists \( K \) such that \( \mathcal{S} \) is \( K \)-synchronizable. Let us show that \( k \leq K \) and \( \mathcal{S} \) is \( k \)-synchronizable.

- \( k \leq K \). Assume by contradiction that \( k \geq K + 1 \). Then there exists exchanges \( \mu_1, \ldots, \mu_n, \mu \) such that \( \mu_1 \cdots \mu_n \cdot \mu \in asTr(\mathcal{S}) \) and \( \mu \) is prime of size \( K + 1 \). Since \( \mu \) is prime, it corresponds to a strongly connected component of size \( K + 1 \) of the conflict graph of \( \mu_1 \cdots \mu_n \cdot \mu \), so \( \mu \) cannot be \( K \)-synchronizable: contradiction.

- \( \mathcal{S} \) is \( k \)-synchronizable. Let \( \mu \in asTr(\mathcal{S}) \) be fixed an let us show that it can be chopped into a sequence of \( k \) exchanges. Since by hypothesis \( \mathcal{S} \) is \( k \)-synchronizable, there are \( K \)-exchanges \( \mu_1, \ldots, \mu_n \) such that \( \mu = \mu_1 \cdots \mu_n \).

Up to decomposing each \( \mu_i \) as a product of prime exchanges, we can assume that all \( \mu_i \) are prime. Moreover, they are all reachable, so their size is bounded by \( k \). As a consequence, \( \mu \) can be decomposed in a sequence of \( k \)-exchanges.

\( \square \)

Since by Theorem [1] it is decidable whether \( \mathcal{S} \) is \( k \)-synchronizable, it is enough to know \( k \) in order to compute \( sd(\mathcal{S}) \). In order to compute \( k \), we have to address two problems: the number of exchanges is possibly infinite, and one should examine sequences of arbitrarily many exchanges. To solve these issues, we are going to reduce to a problem on regular languages. Let \( \Sigma = \{!, \cdot \} \times \mathbb{N} \times \mathbb{N}^2 \); for better readability, we write \( \exists \mathbb{N}^{p+q} \) (resp. \( \exists \mathbb{N}^{p+q} \)) for a \( \Sigma \)-symbol. To every \( \Sigma \)-word \( w \) we associate an MSC \( msc(w) \) as follows. Consider the substitutions \( \sigma_1 : \Sigma \rightarrow S \) and \( \sigma_2 : \Sigma \rightarrow R \cup \{\epsilon\} \) such that \( \sigma_1(!\mathbb{N}^{p+q}) = \sigma_1(!\mathbb{N}^{p+q}) \) = \( \text{send}(p, q, v) \), \( \sigma_2(!\mathbb{N}^{p+q}) = \text{rec}(q, v) \) and \( \sigma_2(!\mathbb{N}^{p+q}) = \epsilon \). Then \( msc(w) \) is defined as \( msc(\sigma_1(w)\sigma_2(w)) \). Clearly, it is an exchange (by construction, it admits a linearization in \( S^* R^* \)), but more remarkably any reachable exchange can be represented by such a word.
Lemma 2. For all reachable exchanges \( \mu \), there exists \( w \in \Sigma^* \) s.t. \( \mu = msc(w) \).

Proof. Let \( \mu \) be a reachable exchange, and let \( \mu_1, \ldots, \mu_n \) be such that \( \mu_1 \cdot \mu_2 \cdots \mu_n \cdot \mu \in asTr(\mathcal{G}) \). There is a linearization of \( \mu_1 \cdots \mu_n \cdot \mu \) which follows the mailbox semantics. This linearization induces a linearization \( lin(\mu) \) of \( \mu \) that also follows the mailbox semantics. Then \( lin(\mu) \) induces an enumeration \( send(p_1, q_1, v_1), \ldots, send(p_n, q_n, v_n) \) of the send events of \( \mu \). Let \( w = a_1 \ldots a_n \) where \( a_i \) is either \( !a_i^p \rightarrow q_i \) if \( send(p_i, q_i, v_i) \) is matched in \( \mu \), or \( ?a_i^p \rightarrow q_i \) if it is unmatched. Then, the claim is that \( msc(w) = \mu \), or in other words, \( \sigma_1(w)\sigma_2(w) \) is a linearization of \( \mu \). By contradiction, assume it is not. Then there are two events \( e, e' \) such that \( e < e' \) in the enumeration \( \sigma_1(w)\sigma_2(w) \) but \( (e', e) \in (\prec_{po} \cup \prec_{src})^* \).

- if \( e, e' \) are two send events then \( e \) occurs before \( e' \) in \( \sigma_1(w) \), i.e. \( e \) occurs before \( e' \) in \( lin(\mu) \), which is a linearization of \( \mu \), and the contradiction with \( (e', e) \in (\prec_{po} \cup \prec_{src})^* \).
- if \( e \) is a send event and \( e' \) a receive event, then \( (e', e) \in (\prec_{po} \cup \prec_{src})^* \) contradicts the fact that \( \mu \) is an exchange.
- if \( e \) is a receive event and \( e' \) is a send event, then \( e < e' \) wrt \( \sigma_1(w)\sigma_2(w) \) contradicts the definition of \( \sigma_1, \sigma_2 \).
- assume finally that \( e \) and \( e' \) are receive events. From \( (e', e) \in (\prec_{po} \cup \prec_{src})^* \), we deduce that \( e' \prec_{po} e \), because \( \mu \) is an exchange. Let \( s, s' \) be the matching send events of \( e, e' \) respectively. Since \( e < e' \) wrt \( \sigma_1(w)\sigma_2(w) \), \( s < s' \) wrt \( \sigma_1(w)\sigma_2(w) \), and therefore \( s < s' \) wrt \( lin(\mu) \). But \( e' < e \) wrt \( lin(\mu) \) because \( e' \prec_{po} e \), which violates the mailbox semantics: contradiction.

\( \square \)

The proof follows from the fact that it is always possible to receive messages in the same global order as they have been sent. Such a property would not hold for peer-to-peer communications, as we can see in the following counter-example.

Consider MSC \( \mu_6 \) on the right. This MSC does not satisfy causal delivery in a mailbox semantics, because the sending of \( v_1 \) happens before the sending of \( v_4 \), and the reception of \( v_4 \) happens before the reception of \( v_1 \). For this reason, there is no word \( w \) such that \( msc(w) \) corresponds to this MSC: such a word would give a linearization that would correspond to a valid mailbox execution. On the other hand, this MSC satisfies causal delivery in a peer-to-peer semantics. For instance, the following linearization is a peer-to-peer execution:

\[ !v_3 !v_4 !v_1 !v_2 ?v_2 ?v_3 ?v_4 ?v_1 \]

We can now define two languages over \( \Sigma \):

\[ \mathcal{L}_1 = \{ w \in \Sigma^* \mid msc(w) \text{ is reachable} \} \text{ and } \mathcal{L}_p = \{ w \in \Sigma^* \mid msc(w) \text{ is prime} \} \]
4 Regularity of reachable exchanges

In this section, we aim at defining a finite state automaton that accepts a word \( w \in \Sigma^* \) iff \( msc(w) \) is reachable, that is, iff there exists \( \mu_1, \mu_2, \ldots, \mu_n \) such that \( \mu_1 \cdot \mu_2 \cdots \cdot \mu_n \cdot msc(w) \in asTr(\mathcal{G}) \). Now, observe that the prefix \( \mu_1 \cdot \mu_2 \cdots \cdot \mu_n \) brings the system in a certain global control state that conditions what can be done by \( msc(w) \). Moreover, the presence of unmatched messages in a buffer imposes that none of the subsequent messages sent to the same buffer can be read.

The construction of the automaton accepting \( \mathcal{L}_r \) proceeds in three separate steps. First, we build an automaton that accepts the language of all words that code one exchange, starting in a certain global control state \( \text{in} \), and ending in another global control state \( \text{fin} \), and possibly not satisfying causal delivery. Secondly, we consider the set of MSCs that satisfy causal delivery. We define automata that recognize the words coding MSCs starting from a certain “buffer state” and ending in another “buffer state”, the “buffer state” characterizing whether or not the MSC satisfies causal delivery. Finally, we show that \( \mathcal{L}_r \) is a boolean combination of the languages of some of these automata.

4.1 Automata of the control states

We consider triples of global states \( (\text{in}, \text{mid}, \text{fin}) \), representing the exchanges such that \( \text{mid} \) can be reached only with sends from \( \text{in} \) and \( \text{fin} \) can be reached only with receptions from \( \text{mid} \). We want to define an automaton \( \mathcal{SR}(\text{in}, \text{mid}, \text{fin}) \) that recognizes the words coding such exchanges. Intuitively, \( \mathcal{SR}(\text{in}, \text{mid}, \text{fin}) \) is a product of on the one hand the global automaton \( \mathcal{G} \) restricted to send transitions and on the other hand \( \mathcal{G} \) restricted to receive transitions. For each send action, either the reception is available and a matched message possible, or there is no corresponding reception and so we obtain an unmatched message.

**Definition 6 (Automaton of control states).** Let \( \mathcal{G} \) be a system and \( \text{in}, \text{mid}, \text{fin} \) global states. \( \mathcal{SR}(\text{in}, \text{mid}, \text{fin}) = (L_{\mathcal{SR}}, \delta_{\mathcal{SR}}, l^0_{\mathcal{SR}}, F_{\mathcal{SR}}) \) is the automaton where:

- \( L_{\mathcal{SR}} = \{ (1, l^1) \mid 1, l^1 \in \mathcal{L}_{\mathcal{G}} \} \); \( l^0_{\mathcal{SR}} = (\text{in}, \text{mid}) \); \( F_{\mathcal{SR}} = \{ (\text{mid}, \text{fin}) \} \);
- for each \( (l_s, \text{send}(p, q, v), l'_s) \in \delta_{\mathcal{G}} \):
  - \( (l_s, 1, !v^{p \rightarrow q}, (l'_s, 1)) \in \delta_{\mathcal{SR}} \) for \( 1 \in \mathcal{L}_{\mathcal{G}} \);
  - if \( (l_r, \text{rec}(q, v), l'_r) \in \delta_{\mathcal{G}} \) then \( ((1, l_r), ?v^{p \rightarrow q}, (l'_s, l'_r)) \in \delta_{\mathcal{SR}} \)

We denote \( \mathcal{L}(\mathcal{SR}(\text{in}, \text{mid}, \text{fin})) \) the language of a such automaton. This is an example of the construction.

**Example 1.** Let \( \mathcal{G}_1 \) be the system whose process automata \( p, q \) and \( r \) are depicted in Fig 4. For the triple \( (\text{in}, \text{mid}, \text{fin}) \) where \( \text{in} = (0, 0, 0) \), \( \text{mid} = (2, 0, 1) \) and
\[
\mathcal{L}(\text{SR}(\text{in}, \text{mid}, \text{fin})) = \text{!}??a^{p \rightarrow r}(\text{!}??c^{p \rightarrow q} \text{!}??b^{r \rightarrow q} + \text{!}??b^{r \rightarrow q} \text{!}??c^{p \rightarrow q}) + \text{!}??b^{r \rightarrow q} \text{!}??a^{p \rightarrow r} \text{!}??c^{p \rightarrow q}
\]

\[
\begin{array}{c}
\text{fin} = (2, 1, 2), \text{automaton } \text{SR}(\text{in}, \text{mid}, \text{fin}) \text{ is depicted below the system and has for language:}
\end{array}
\]

\[
\mathcal{L}(\text{SR}(\text{in}, \text{mid}, \text{fin})) = !?a^{p \rightarrow r}(!?c^{p \rightarrow q}!?b^{r \rightarrow q} + !?b^{r \rightarrow q}!?c^{p \rightarrow q}) + !?b^{r \rightarrow q}!?a^{p \rightarrow r}!?c^{p \rightarrow q}
\]

**Fig. 4.** System \( \mathcal{E}_1 \) and Automaton \( \text{SR}(\text{in}, \text{mid}, \text{fin}) \)

**Lemma 3.** \( w \in \mathcal{L}(\text{SR}(\text{in}, \text{mid}, \text{fin})) \) for some \( \text{mid} \) iff \( w \in \text{usc}(w) \) \( \text{fin} \).

**Proof.** Observe that, by construction of \( \text{SR}(\text{in}, \text{mid}, \text{fin}) \), \( l^0_{\text{SR}} \xrightarrow{w} (1,1') \) iff \( \sigma_1(w) \xrightarrow{\text{usc}(w)} \text{fin} \) and \( \text{mid} \xrightarrow{\sigma_2(w)} \text{fin} \) (this can be shown by an easy induction on the length of \( w \)). In particular, \( w \) is accepted iff \( w \xrightarrow{\sigma_1(w)} \text{mid} \) and \( \text{mid} \xrightarrow{\sigma_2(w)} \text{fin} \), which is equivalent to \( w \xrightarrow{\text{usc}(w)} \text{fin} \). \( \square \)

### 4.2 Automata of causal delivery exchanges

Let us now move to the trickier part, namely the recognition of words coding MSCs that satisfy causal delivery. Let \( \mu = (E_v, \lambda, \prec_{po}, \prec_{src}) \) be an MSC, and \( v \in \lambda^{-1}(S) \) a send event, we write \( ev_S(v) \) for the event \( v \) and, when it exists, \( ev_R(v) \) for the event \( v' \in \lambda^{-1}(R) \) such that \( v \prec_{src} v' \). We say that \( v \) is unmatched
The extended conflict graph is acyclic (Theorem 2 in [7]). Moreover, we write $B$ contains enough information to determine whether its extended conflict graph is acyclic. We write $B$ is matched $\forall v, v' \in \{X,Y\}$ such dependencies between events. The figure on the right represents an MSC and its associated conflict graph.

**Definition 7 (Conflict Graph).** The conflict graph $CG(\mu)$ of an MSC $\mu = (E, \lambda, \prec_{po}, \prec_{src})$ is the labeled graph $(V, \{XY\}_{X,Y \in \{R,S\}^*})$ where $V = \lambda^{-1}(S)$, and for all $v, v' \in V$, there is a $XY$ dependency edge $v \overset{XY}{\rightarrow} v'$ between $v$ and $v'$ $(X,Y \in \{S,R\})$, if $ev_X(v)$ and $ev_Y(v')$ are defined and $ev_X(v)$ $\prec_{po}$ $ev_Y(v')$.

The extended conflict graph $ECG(\mu)$ is obtained by adding all dashed edges $v \overset{XY}{\rightarrow} v'$ satisfying the relation $\overset{XY}{\rightarrow}$ in Fig. 6. Intuitively, $v \overset{XY}{\rightarrow} v'$ expresses that the event $X$ of $v$ must happen before the event $Y$ of $v'$ due to: their order on the same machine (Rule 1), or the fact that a send happens before its matching receive (Rule 2), or to the mailbox semantics (Rules 3 and 4), or because of a chain of such dependencies (Rule 5). This captures all constraints induced by the mailbox communication, and it has been shown that an MSC satisfies causal delivery if and only if its extended conflict graph is acyclic (Theorem 2 in [7]).

We build an automaton that recognizes the words $w$ such that $msc(w)$ satisfies causal delivery. To this aim, we associate to each MSC a “buffer state” that contains enough information to determine whether its extended conflict graph is acyclic. We write $\mathbb{B}$ for the set $(2^\mathbb{P} \times 2^\mathbb{P})^\mathbb{P}$. The buffer state $B(\mu) \in \mathbb{B}$ of the MSC $\mu$ is the tuple $B(\mu) = (C_{S,p}^{\mu}, C_{R,p}^{\mu})_{p \in \mathbb{P}}$ such that for all $p \in \mathbb{P}$:

$$
C_{S,p}^{\mu} = \{\text{proc}_S(v) \mid v' \overset{SS}{\rightarrow} v \& v' \text{ is unmatched} \& \text{proc}_R(v') = p\} \cup \{\text{proc}_S(v) \mid v \text{ is unmatched} \& \text{proc}_R(v) = p\}
$$

$$
C_{R,p}^{\mu} = \{\text{proc}_R(v) \mid v' \overset{SS}{\rightarrow} v \& v' \text{ is unmatched} \& \text{proc}_R(v') = p \& v \text{ is matched}\}
$$

We can show that the $ECG(\mu)$ is acyclic if for all $p \in \mathbb{P}$, $p \not\in C_{R,p}^{\mu}$ (immediate consequence of Theorem 2 in [7]). Moreover, we write $\mathbb{B}_{good}$ for the subset of $\mathbb{B}$ formed by the tuples $(C_{S,p}^{\mu}, C_{R,p}^{\mu})_{p \in \mathbb{P}}$ such that $p \not\in C_{R,p}^{\mu}$ for all $p$.

**Proposition 1 ([7]).** For $w \in \Sigma^*$, $msc(w)$ satisfies causal delivery if and only if $B(\mu(w)) \in \mathbb{B}_{good}$. 

![Fig. 5. MSC $\mu_3$ and its conflict graph](image)

![Fig. 6. Deduction rules for extended dependency edges of the conflict graph](image)
Noticing that $\mathbb{B}$ is finite, we build an automaton $A(B_0, B_1)$ with $B_0, B_1 \in \mathbb{B}$. The intuition behind these two buffer states is that $B_0$ summarises the conflict graph derived from previous exchanges and $B_1$ summarises the conflict graph obtained when a new exchange is added.

**Definition 8 (Automaton of causal exchanges).** The automaton $A(B_0, B_1)$ is defined as follows:

- $\mathbb{B}$ is the set of states,
- $B_0$ is the initial state (hereafter, we assume that $B_0 = (C_{S,p}^{(0)}, C_{R,p}^{(0)})_{p \in P}$).
- $\{B_1\}$ is the set of final states
- the transition relation $(\tau_a)_{a \in \Sigma}$ is defined as follows:
  - $(C_{S,p}, C_{R,p})_{p \in P} \xrightarrow{?_{e\rightarrow-q}} (C'_{S,p}, C'_{R,p})_{p \in P}$ holds if for all $r \in P$: let the intermediate set $C_{S,r}''$ be defined by
    \[ C_{S,r}'' = \begin{cases} C_{S,r} \cup \{p\} & \text{if } p \in C_{R,r}^{(0)} \text{ or } q \in C_{R,r} \\ C_{S,r} \text{ otherwise} & \end{cases} \]
    Then\n    \[ C'_{S,r} = \begin{cases} C''_{S,r} \cup C_{S,q} & \text{if } p \in C''_{S,r} \text{ and } C'_{R,r} = \begin{cases} C_{R,r} \cup \{q\} \cup C_{R,q} & \text{if } p \in C''_{S,r} \\ C_{R,r} \text{ otherwise} & \end{cases} \]
  - $(C_{S,p}, C_{R,p})_{p \in P} \xrightarrow{?_{e\rightarrow-q}} (C'_{S,p}, C'_{R,p})_{p \in P}$ holds if for all $r \in P$,
    \[ C'_{S,r} = \begin{cases} C_{S,r} \cup \{p\} & \text{if } q = r \text{ or } q \in C_{R,r} \\ C_{S,r} \text{ otherwise} & \end{cases} \text{ and } C'_{R,r} = C_{R,r} \]

Let $L(B_0, B_1)$ denote the language recognized by $A(B_0, B_1)$.

**Example 2.** Consider $\mu_4 = msc(w)$ with $w = !_v_{3}^{p_1 } ^{p_2 } ^{p_4 } ?_{e_4 } !v_{5} ^{p_4 } ^{p_6 } ?_{e_6 } !v_{6} ^{p_6 }$ and assume we start with $B_0$ such that $C_{S,p_5} = \{ p_4 \}$ and $C_{R,p_5} = \{ p_3 \}$. Then the update of $B$ (or, more precisely, of $C_{S,p_5}$, $C_{R,p_5}$, and $C_{S,p_2}$) after reading each message is shown below. Note how $v_6$ has no effect, despite the fact that $p_6 \in C_{R,p_5}$ at the time the message is read.

\[
\begin{align*}
C_{S,p_5} &\{ p_4 \} & \xrightarrow{?_{e_3}} & \{ p_4 \} & \xrightarrow{?_{e_4}} & \{ p_1, p_3, p_4 \} & \xrightarrow{?_{e_5}} & \{ p_1, p_3, p_4 \} & \xrightarrow{?_{e_6}} & \{ p_1, p_3, p_4 \} \\
C_{R,p_5} &\{ p_3 \} & \xrightarrow{?_{e_3}} & \{ p_3 \} & \xrightarrow{?_{e_4}} & \{ p_2, p_3 \} & \xrightarrow{?_{e_5}} & \{ p_2, p_3, p_6 \} & \xrightarrow{?_{e_6}} & \{ p_2, p_3, p_6 \} \\
C_{S,p_2} &\emptyset & \xrightarrow{?_{e_3}} & \{ p_1 \} & \xrightarrow{?_{e_4}} & \{ p_1 \} & \xrightarrow{?_{e_5}} & \{ p_1 \} & \xrightarrow{?_{e_6}} & \{ p_1 \}
\end{align*}
\]

Next lemma states that $A(B, B')$ recognizes the words $w$ such that $msc(w)$, starting with an initial buffer state $B$, ends in final buffer state $B'$. 
Lemma 4. Let \( B, B' \in \mathbb{B} \) and \( w \in \Sigma^* \). Then \( w \in \mathcal{L}(B, B') \) if and only if for all MSC \( \mu \) such that \( B = B(\mu), B' = B(\mu \cdot msc(w)) \).

Proof. Take \( w = a_0 \ldots a_n \in \Sigma^* \). To prove the lemma it is sufficient to show that if \( B = B(\mu) \) and :

\[
B = (C_{S,p}^{(0)}, C_{R,p}^{(0)})_{p \in \mathbb{P}} \xrightarrow{a_0} (C_{S,p}^{(1)}, C_{R,p}^{(1)})_{p \in \mathbb{P}} \xrightarrow{a_1} \ldots \xrightarrow{a_n} (C_{S,p}^{(n+1)}, C_{R,p}^{(n+1)})_{p \in \mathbb{P}} = B',
\]

then \( B' = B(\mu \cdot msc(w)) \).

The proof proceeds by induction on the length of \( w \) where the inductive hypothesis is that \((C_{S,p}^{(n)}, C_{R,p}^{(n)})_{p \in \mathbb{P}} = B(\mu \cdot msc(a_0 \ldots a_{n-1})) \).

We start by showing that \( \forall r \in \mathbb{P}, C_{S,r}^{(n+1)} = C_{S,r}^{\mu-msc(w)} \). Suppose that \( p \in C_{S,r}^{(n+1)} \). If \( p \in C_{S,r}^{(n)} \) then we can immediately conclude that \( p \in C_{S,r}^{\mu-msc(w)} \) since \( C_{S,r}^{(n)} = C_{S,r}^{\mu} \subseteq C_{S,r}^{(n+1)} \) and function \( B(\cdot) \) is increasing monotone. Instead if \( p \notin C_{S,r}^{(n)} \) then the following can happen (without loss of generality suppose that \( p \) has been added while reading the last symbol of \( w \)):

1. \( a_n = \mu v^{p-q} \) and \( p \in C_{S,r}^{(0)} = C_{R,r}^{\mu} \)
   Then there exists a message in \( \mu \), \( v'' \) such that \( \text{proc}_R(v'') = p \) and \( v' \xrightarrow{SS} v'' \) with \( v' \) unmatched. Then it is easy to see that \( v'' \xrightarrow{SS} v \) and therefore \( p \in C_{S,r}^{\mu-msc(w)} \);
2. \( a_n = ?v^{p-q} \) and \( q \in C_{S,r}^{(n)} = C_{S,q}^{\mu-msc(a_0 \ldots a_{n-1})} \)
   Then there exists a message \( v'' \) in \( \mu \cdot msc(a_0 \ldots a_{n-1}) \), such that \( \text{proc}_R(v'') = q \) and \( v' \xrightarrow{SS} v'' \) with \( v' \) unmatched. Then it is easy to see that \( v'' \xrightarrow{SS} v \) and therefore \( p \in C_{S,r}^{\mu-msc(w)} \);
3. \( a_n = \mu v^{p-q} \) and \( p' \in C_{S,r}^{(n)} \) and \( p \in C_{S,q}^{(n)} = C_{S,q}^{\mu-msc(a_0 \ldots a_{n-1})} \)
   Then there exists a message \( v'' \) in \( \mu \cdot msc(a_0 \ldots a_{n-1}) \), such that \( \text{proc}_R(v'') = \mu v^{p'} \) and \( v' \xrightarrow{SS} v'' \) with \( v' \) unmatched and \( \text{proc}_R(v') = q \). Then it is easy to see that \( v \xrightarrow{SS} v'' \) and since \( p' \in C_{S,r}^{(n)} \) and with an analysis similar to the one above we have \( v'' \xrightarrow{SS} v \) with \( v'' \) unmatched and we can conclude \( p \in C_{S,r}^{\mu-msc(w)} \).
4. \( a_n = ?v^{p-q} \) and \( p \in C_{R,r} \)
   Analogous to case 2 above.
5. \( a_n = \mu v^{p-r} \)
   We can immediately conclude \( p \in C_{S,r}^{\mu-msc(w)} \) as \( v \) is unmatched and \( \text{proc}_R(v) = r \).

Now suppose that \( p \in C_{S,r}^{\mu-msc(w)} \) (without loss of generality we can assume \( p \notin C_{S,r}^{\mu-msc(a_0 \ldots a_{n-1})} \)). Then either \( a_n = \mu v^{p-q} \) then it is immediate to see that \( p \in C_{S,r}^{(n+1)} \) or \( a_n = ?v^{p-q} \) and \( v' \xrightarrow{SS} v \) for some \( v' \) unmatched and \( \text{proc}_R(v') = r \).

To prove the lemma it is sufficient to show that \( B = B(\mu) \) and :

\[
B = (C_{S,p}^{(0)}, C_{R,p}^{(0)})_{p \in \mathbb{P}} \xrightarrow{a_0} (C_{S,p}^{(1)}, C_{R,p}^{(1)})_{p \in \mathbb{P}} \xrightarrow{a_1} \ldots \xrightarrow{a_n} (C_{S,p}^{(n+1)}, C_{R,p}^{(n+1)})_{p \in \mathbb{P}} = B',
\]

then \( B' = B(\mu \cdot msc(w)) \).
that since \( w \) is an exchange, \( p \notin C_{R,r}^{\mu \cdot \text{msc}(a_0 \ldots a_{n-1})} \setminus C_{R,r}^{\mu} \). In both cases we can conclude \( p \in C_{S,r}^{(n+1)} \).

Next we show that \( \forall r \in \mathbb{P}, C_{R,r}^{n} = C_{R,r}^{\mu \cdot \text{msc}(w)} \). Suppose that \( p \in C_{R,r}^{(n+1)} \) (without loss of generality we can assume \( p \notin C(n+1)_{R,r} \)). This entails that:

- either \( a_n = ?! ? v^q \rightarrow p \) with \( q \in C_{S,r}^{(n+1)} = C_{S,r}^{\mu \cdot \text{msc}(w)} \); then there exists a message \( v'' \) in \( \mu \cdot \text{msc}(w) \), such that \( \text{proc}_S(v'') = q \) and \( v'' \rightarrow v' \) with \( v' \) unmatched and \( \text{proc}_R(v') = r \). Then it is easy to see that \( v'' \rightarrow v \) and we can conclude \( p \in C_{R,r}^{\mu \cdot \text{msc}(w)} \);
- or \( a_n = ?! ? v^q \rightarrow p \) with \( q \in C_{S,r}^{(n+1)} = C_{S,r}^{\mu \cdot \text{msc}(w)} \) and \( p \in C_{R,p'}^{(n+1)} = C_{R,p'}^{\mu \cdot \text{msc}(a_0 \ldots a_{n-1})} \); then there exists a message \( v'' \) in \( \mu \cdot \text{msc}(a_0 \ldots a_{n-1}) \), such that \( \text{proc}_S(v'') = q \) and \( v'' \rightarrow v' \) with \( v' \) unmatched and \( \text{proc}_R(v') = r \). Similarly there is \( v''' \) in \( \mu \cdot \text{msc}(a_0 \ldots a_{n-1}) \) such that \( \text{proc}_R(v''') = p \) and \( v'''' \rightarrow v' \) with \( v'''' \) unmatched and \( \text{proc}_R(v''') = p' \). Now when adding \( v \) to the conflict graph we have \( v' \rightarrow v \). Hence we can conclude \( p \in C_{R,r}^{\mu \cdot \text{msc}(w)} \).

Now suppose that \( p \in C_{R,r}^{n} \) (without loss of generality we can assume \( p \notin C_{R,r}^{\mu \cdot \text{msc}(a_0 \ldots a_{n-1})} \)). We know \( C_{R,r}^{\mu \cdot \text{msc}(a_0 \ldots a_{n-1})} = C_{R,r}^{(n)} \) and let \( a_n = ?! ? v^q \rightarrow p \).

The following can happen: \( q \in C_{S,r}^{\mu \cdot \text{msc}(a_0 \ldots a_{n-1})} \), or \( q \in C_{R,r}^{\mu \cdot \text{msc}(a_0 \ldots a_{n-1})} \), or \( p \in C_{S,r}^{\mu \cdot \text{msc}(a_0 \ldots a_{n-1})} \). The last case is when \( a_n = ?! ? v^q \rightarrow p' \) and \( p \in C_{R,p'}^{\mu \cdot \text{msc}(a_0 \ldots a_{n-1})} \) and \( q' \in C_{S,r}^{\mu \cdot \text{msc}(a_0 \ldots a_{n-1})} \). For all this cases, by inductive hypothesis and by Definition S we can conclude \( p \in C_{S,r}^{(n+1)} \).

\[ \square \]

### 4.3 Language of reachable exchanges

The only thing that remains to do is to combine the previous automata to define one that recognizes the (codings of) reachable exchanges. The language \( \mathcal{L}(\text{SR(in, mid, fin)}) \) contains arbitrary exchanges which do not necessarily satisfy causal delivery. Here comes into play the \( A(B, B') \) automata, where we take \( B \) and \( B' \in \mathbb{B}_{\text{good}} \) in order to ensure causal delivery.

Let

\[
\mathcal{L}_C(\text{in, fin, } B, B') \overset{\text{def}}{=} \bigcup_{\text{mid} \in L_C} \mathcal{L}(\text{SR(in, mid, fin)}) \cap \mathcal{L}(B, B').
\]

Intuitively, \( \mathcal{L}_C(\text{in, fin, } B, B') \) is the language of (codings of) exchanges between global states \( \text{in} \) and \( \text{fin} \) starting with an initial buffer state \( B \) and ending in final buffer state \( B' \); when moreover \( B, B' \in \mathbb{B}_{\text{good}} \), these exchanges satisfy causal delivery.

The last step is to combine causal delivery exchanges so that they can be performed by the system one after the other from the initial state \( I_0 \). This motivates the definition of the following set \( R \) of reachable languages. Let \( B_\emptyset = (\emptyset, \emptyset)_{p \in \mathbb{P}} \).
Definition 9 (Reachable languages). Given a system $\mathcal{S} = (L_\mathcal{S}, \delta_\mathcal{S}, l_0)$, the set $\mathcal{R}$ of reachable languages is the least set of languages of the form $\mathcal{L}_c(\text{in}, \text{fin}, B, B_f)$ defined as follows.

1. for any $l \in L_\mathcal{S}$ and $B \in \mathbb{B}_{\text{good}}$, $\mathcal{L}_c(l_0, l, B, B_f) \in \mathcal{R}$
2. for any $l_1, l_2, l_3 \in L_\mathcal{S}$ and any $B_1, B_2, B_3 \in \mathbb{B}_{\text{good}}$, if $\mathcal{L}_c(l_1, l_2, B_1, B_2) \in \mathcal{R}$ and $\mathcal{L}_c(l_1, l_3, B_1, B_2) \neq \emptyset$ then $\mathcal{L}_c(l_2, l_3, B_2, B_3) \in \mathcal{R}$.

Then the union $\bigcup \mathcal{R}$ of all reachable languages is equal to the language $\mathcal{L}_r = \{w \in \Sigma^* \mid \text{msc}(w) \text{ is reachable}\}$. As a consequence, we get the following result.

Theorem 2. $\mathcal{L}_r$ is a regular language and is accepted by an effective finite state automaton.

Proof. $\Rightarrow$ $w \in \bigcup \mathcal{R}$ so there is a sequence of words $w_1, \ldots, w_n \in \Sigma^*$ such that for all $1 \leq j \leq n$, $w_j \in \mathcal{L}(l_{j-1}, l_j)$ and $l_j \in \mathbb{B}_{\text{good}}$. So, each $\text{msc}(w_j)$ verifies causal delivery, and, we can easily show by induction that $\text{msc}(w_1) \cdot \text{msc}(w_2) \cdot \ldots \cdot \text{msc}(w_n) \cdot \text{msc}(w) \in \text{asTr}(\mathcal{S})$ and so, $\text{msc}(w)$ is reachable.

$\Leftarrow$ $w \in \mathcal{L}(\text{sr}(l_{j-1}, l_j))$ for $l_j \in L_\mathcal{S}$, and, by Lemma 4, $l_{j-1} \xrightarrow{\text{msc}(w)} l_j$, and $l_n \xrightarrow{\text{msc}(w)} \text{fin}$. Moreover, for all $1 \leq j \leq n$, $w_j \in \mathcal{L}(B_{j-1}, B_j)$ and $B_j \in \mathbb{B}_{\text{good}}$. So, each $\text{msc}(w_j)$ verifies causal delivery, and, we can easily show by induction that $\text{msc}(w_1) \cdot \text{msc}(w_2) \cdot \ldots \cdot \text{msc}(w_n) \cdot \text{msc}(w) \in \text{asTr}(\mathcal{S})$ and so, $\text{msc}(w)$ is reachable.

5 Prime exchanges

We reformulate the primality of an exchange in terms of its conflict graph. We say that the conflict graph $\mathbb{G}(\mu)$ associated with the MSC $\mu$ is strongly connected if for all $v, v' \in V$, it holds that $v \rightarrow^* v'$, where $\rightarrow^*$ is the reflexive transitive closure of $\rightarrow = \bigcup_{X,Y \in \{S, R\}} \xrightarrow{XY}$. 
Lemma 5. An exchange \( \mu \) is prime iff \( \text{CG}(\mu) \) is strongly connected.

Proof. Let \( \mu = \mu_1 \cdots \mu_n \) be an MSC formed with a sequence of exchanges. Let \( e, e' \) be two events of \( \mu \), and let \( i, i' \in \{1, \ldots, n\} \) be such that \( e \) appears in \( \mu_i \) and \( e' \) appears in \( \mu_{i'} \). If there is an edge \( e \xrightarrow{XY} e' \) in the conflict graph of \( \mu \), then \( i \leq i' \). As a consequence, if \( e \) and \( e' \) are on a same strongly connected component, then \( i = i' \), and if the conflict graph of \( \mu \) is strongly connected, then \( n = 1 \) and \( \mu \) is a prime exchange. \( \square \)

Next we discuss the construction of the automaton that recognizes \( \{ w \in \Sigma^* \mid \text{msc}(w) \text{ is prime} \} \). Since there are infinitely many \( \text{CG}(\text{msc}(w)) \), in order to have a finite state automaton, we compute a finite abstractions of \( \text{CG}(\text{msc}(w)) \) that is sound in the sense that \( \text{CG}(\text{msc}(w)) \) is strongly connected if and only if its abstraction is of a certain shape. Let us now define this abstraction.

We need to define some graph transformations. The graphs we are going to manipulate are oriented graphs labeled with a pair of set of processes on each vertex. We call such objects P-graphs. Formally, a P-graph is a tuple \((V, E, \lambda)\) with \( E \subseteq V \times V \) and \( \lambda_V : V \to 2^\Sigma \) for \( x \in \{S, R\} \). The P-graph \( \text{pgr}(\mu) \) associated with the conflict graph \( \text{CG}(\mu) = (V, \{ \xrightarrow{XY} \}_{X,Y \in \{S,R\}}) \) is \((V, E, \lambda_S, \lambda_R)\) where (1) \((v, v') \in E \) if \( v \xrightarrow{XY} v' \) for some \( X, Y \), (2) \( \lambda_S(v) = \{ \text{proc}_S(v) \} \), and (3) \( \lambda_R(v) = \{ \text{proc}_R(v) \} \), and if \( v \) is unmatched \( \lambda_R(v) = \emptyset \).

The first graph transformation we consider consists in merging the vertices that belong to a same strongly connected component (SCC). Formally, let \( G = (V, E, \lambda_S, \lambda_R) \) be a P-graph, and let \( \text{merge}(G) = (V', E', \lambda_S, \lambda_R) \) be defined by (1) \( V' \) is the set of maximal SCCs of \( G \), (2) for two distinct maximal SCCs \( U, U' \), \((U, U') \in E' \) if there are \( v \in U \) and \( v' \in U' \) such that \( (v, v') \in E^+ \) (the transitive closure of \( E \)), (3) for \( X = S, R \), \( \lambda_X(U) = \bigcup_{v \in U} \lambda_X(v) \).

The second graph transformation we consider consists in erasing some of the processes that appear in the labels. Let \( G = (V, E, \lambda_S, \lambda_R) \) be a fixed P-graph, and let \( v \in V, X \in \{S, R\}, \) and \( p \in \lambda_X(v) \) be fixed. We say that \( p \) is \( X \)-redundant in \( v \) if there are \( v_1, v_2 \) such that (1) \((v_1, v) \in E^+ \) and \((v, v_2) \in E^+ \), and (2) \( p \in \lambda_X(v_1) \cap \lambda_X(v_2) \). Intuitively, \( p \) is redundant in \( v \) if it also appears in the label of an ancestor and a descendant of \( v \). We define the P-graph \( \text{erase}(G) \) as \((V, E, \lambda'_S, \lambda'_R)\) where for all \( X \in \{S, R\}, \) for all \( v \in V, \lambda'_X(v) \) is the set of processes \( p \in \lambda_X(v) \) such that \( p \) is not \( X \)-redundant at \( v \).

The last graph transformation we consider consists in sweeping out the vertices labeled with empty sets of processes. Formally, for \( G = (V, E, \lambda_S, \lambda_R) \), the P-graph \( \text{pgr}(G) = (V', E', \lambda_S, \lambda_R) \) where \( V' = \{ v \in V \mid \lambda_S(v) \cup \lambda_R(v) \neq \emptyset \} \) and \( E' = E \cap V' \times V' \). The abstraction \( \alpha(G) \) of a P-graph \( G \) is defined as \( \text{pgr}(\text{pgr}(G)) \). An example of the construction is in Fig \( \text{Fig.} \)

Lemma 6. \( \text{CG}(\mu) \) is strongly connected iff \( \alpha(\text{pgr}(\mu)) \) is a single vertex graph.

Proof. By definition of \( \alpha \), and in particular of function \( \text{merge}(\cdot) \), a vertex of \( \alpha(\text{pgr}(\mu)) \) corresponds to a strongly connected component of \( \text{CG}(\mu) \). \( \square \)
Fig. 8. MSC $\mu_5$, its associated P-graph $pgr(\mu_5)$, and the abstraction $\alpha(pgr(\mu_5))$.

By construction, for any process $p$, and for any $X \in \{S, R\}$, there are at most two vertices $v$ of $\alpha(pgr(\mu))$ such that $p \in \lambda_X(v)$. From this, we deduce that $\alpha(pgr(\mu))$ has at most $2|P|$ vertices, and as a consequence:

**Lemma 7.** $\sharp\{\alpha(pgr(\mu)) \mid \mu \text{ is an exchange}\} \leq 2^6|P|^2$.

**Proof.** Let $n \geq 0$ be fixed and let us give an upper bound on the number of P-graphs with $n$ vertices. First, there are $2^n(n-1)$ different possible choices for the edge relation. By construction (in particular, by definition of function $\text{erase}(.)$), it holds that

$$ (1) \quad \forall p \in P, \forall X \in \{S, R\}, \sharp\{v \mid p \in \lambda_X(v)\} \leq 2. $$

A choice for the $\lambda$ function is therefore the choice, for each $p$, of at most two vertices $v$ such that $p \in \lambda_S(v)$ and at most two other vertices $v$ such that $p \in \lambda_R(v)$. So there are at most $n^4$ different choices for each $p$, and at most $n^4|P|$ different choices for $\lambda$. To sum up, there are less than $2^n n^4|P|$ P-graphs with $n$ vertices.

Now, again from (1), there are at most $2|P|$ vertices in a P-graph, so the number of P-graph is bounded by

$$ \sum_{n=1}^{2|P|} 2^n n^4|P| \leq 2|P|2^{|P|^2}(2|P|)^4|P| \leq 2|P|^2 2^{|P|^2}(2|P|)^4|P| = 2^6|P|^2. $$

□

There are therefore finitely many $\alpha(pgr(\mu))$. This allows us to define the automaton that computes $\alpha(pgr(msc(w)))$ for any $w \in \Sigma^*$ and accepts $w$ in the language of this new automaton when this P-graph is a single vertex graph. Let $G = (V, E, \lambda_S, \lambda_R)$ and a letter $\ddagger v^p q^r \in \Sigma$ be fixed. We want to define the transition function $\delta_g$ of our automaton, or in other words, the P-graph $\delta_g(G, \ddagger v^p q^r)$
reached after adding the message $\dagger v^{p\rightarrow q}$ to the MSC. We let $\delta_g(G, \dagger v^{p\rightarrow q}) = \alpha(G')$. $G' = (V', E', \lambda'_S, \lambda'_R)$ is defined as follows: (1) $V' = V \cup \{v_0\}$, (2) $\lambda'_S(v_0) = \{p\}$, (3) if $\dagger = \triangleright?$, then $\lambda'_R(v_0) = \{q\}$, and if $\dagger = \triangleright$, then $\lambda'_R(v_0) = \emptyset$. (4) for all $v \in V$, for all $X \in \{S, R\}$, $\lambda'_X(v) = \lambda_X(v)$, and (5) the set of edges $E'$ is defined as

$$E' = E \cup \{(v, v_0) \mid p \in \lambda_S(v)\} \cup \{(v_0, v) \mid p \in \lambda_R(v)\} \cup \{(v, v_0) \mid q \in \lambda_S(v) \cup \lambda_R(v)\} \text{ if } \dagger = \triangleright?$$

$$\emptyset \text{ if } \dagger = \triangleright.$$  

For example, consider the MSC $\mu$ of Fig. 8 and let $G = \alpha(pgr(\mu))$ be its associated abstracted P-graph. Let $G'$ be defined as above while reading $!\triangleright v^{p\rightarrow q}_6$. Then $G'$ is the graph on the right, and $\delta_g(G, v^{p\rightarrow q}_6)$ is a single vertex graph.

**Lemma 8.** $\delta_g(\alpha(pgr(msc(w))), \dagger v^{p\rightarrow q}) = \alpha(pgr(msc(w \cdot \dagger v^{p\rightarrow q}))).$

Before we prove Lemma 8, we need to introduce a few notions and observations. Let $G = (V, E, \lambda_S, \lambda_R)$ be a P-graph. A vertex $v \in V$ is X-covered if for all $p \in \lambda_X(v)$ $p$ is X-redundant. We also say that $v \in V$ is covered if it is both S-covered and R-covered. A partial abstraction of $G$ is a graph $G' = (V', E', \lambda'_S, \lambda'_R)$ such that

- $V' = \{V_1, \ldots, V_n\}$ where each $V_i$ is a (not necessarily maximal) strongly connected component of $G$, all $V_i$ are disjoint, and for all $v \in V \setminus \bigcup_{i=1}^n V_i$, $v$ is covered.
- for all $i, j$, $(V_i, V_j) \in E'$ iff $(v, v') \in E$ for some $v \in V_i$ and some $v' \in V_j$.
- for all $i, X$, $\lambda_X(V_i) = \bigcup_{v \in V_i} \lambda_X(v)$

Intuitively, $G'$ is a partial abstraction of $G$ if it results from a “partial application” of the functions $merge(\cdot)$, $erase(\cdot)$, and $sweep(\cdot)$: some vertices of a same SCC are merged, but not necessarily all, some labels are erased, but not necessarily all, and some vertices are swept, but not necessarily all. From this observation, it follows the following: if $G'$ is a partial abstraction of $G$, then $\alpha(G') = \alpha(G)$.

**Proof.** Let $w \in \Sigma^*$ and $\dagger v^{p\rightarrow q}$ be fixed. Let $G_1 = pgr(msc(w))$ and $G_2 = pgr(msc(w \cdot \dagger v^{p\rightarrow q}))$ and let us compare $G_1$ and $G_2$. First, there is an extra vertex $v_0$ in $G_2$ that represents $\dagger v^{p\rightarrow q}$, with $\lambda_S(v_0) = \{p\}$ and either $\lambda_R(v_0) = \{q\}$ (if $\dagger = \triangleright?$) or $\lambda_R(v_0) = \emptyset$ (if $\dagger = \triangleright$). Now, consider the extra edges. Obviously, these extra edges have $v_0$ either as source or as destination. First consider the edges with $v_0$ as destination. The send event of $v_0$ happens after all send events of $p$, so for all $v \neq v_0$ such that $p \in \lambda_S(v)$, $(v, v_0) \in E_2$. In the case where $\dagger = \triangleright?$, the receive event of $v_0$ also happens after all send and receive events of $q$, so for all $v \neq v_0$ such that $q \in \lambda_S(v) \cup \lambda_R(v)$, $(v, v_0) \in E_2$. There are no other incoming edges in $v_0$ now, consider the outgoing edges of $v_0$. The send event of $v_0$ happens before all receive events of $p$, so for all $v \neq v_0$ such that $p \in \lambda_R(v)$, $(v_0, v) \in E_2$. To sum up, we have:
\[ E_2 = E_1 \cup \{(v, v_0) \mid p \in \lambda_S(v)\} \]
\[ \cup \{(v_0, v) \mid p \in \lambda_R(v)\} \]
\[ \cup \begin{cases} \{(v, v_0) \mid q \in \lambda_S(v) \cup \lambda_R(v)\} & \text{if } \dagger = !? \\ \emptyset & \text{if } \dagger = ! \end{cases} \]

Observe now that the rules to add vertices and edges to go from $G_1$ to $G_2$ are exactly the same as the rules to go from $G$ to $G'$ in the definition of $\delta_g(G, \dagger v^p \rightarrow q)$. Assume that $G = \alpha(G_1) = \alpha(pgr(msc(w))$. Then $G'$ is a partial abstraction of $G_2$. So by the discussion above,

\[ \alpha(G') = \alpha(G_2). \]

Now, by definition of $\delta_g$, $\delta_g(G, \dagger v^p \rightarrow q) = \alpha(G')$. To sum up

\[ \delta_g(G, \dagger v^p \rightarrow q) = \alpha(G_2) = \alpha(pgr(msc(w \cdot \dagger v^p \rightarrow q))). \]

\[ \Box \]

**Theorem 3.** There is an effective deterministic finite state automaton $A$ with less than $2^{6|P|^2}$ states such that $L(A) = \{w \in \Sigma^* \mid msc(w) \text{ is prime}\}$.

**Proof.** Let $A = (Q, \Sigma, \delta, q_0, F)$ be defined by

- $Q = \{\alpha(pgr(\mu)) \mid \mu \text{ is an exchange}\}$;
- $\delta$ as defined in Section 5;
- $q_0 = \alpha(pgr(\epsilon))$ where $\epsilon$ denotes the empty MSC;
- $F = \{G \in Q \mid |G| = 1\}$

Then by Lemma 7, $A$ is a deterministic finite state automaton with at most $2^{6|P|^2}$ states. Moreover, by Lemma 8 for all $w$, $\delta_g(q_0, w) = pgr(msc(w))$, so $w$ is accepted iff $pgr(msc(w))$ is a single vertex graph. By Lemma 6 this is equivalent to the fact that $msc(w)$ is prime. $\Box$

### 6 Computation of $k$

So far we have shown: in Lemma 1, we established that a way to compute $sd(G)$ was to compute the length $k$ of the largest prime reachable exchange. To every word $w \in \Sigma^*$, we associated an MSC $msc(w)$, and we showed that for every reachable MSC $\mu$, there exists a word $w \in \Sigma^*$ such that $\mu = msc(w)$ (Lemma 2). We deduced that $k$ corresponds to the length of the longest word of $L_T \cap L_P$, if $L_T \cap L_P$ is finite, otherwise $k = \infty$. In Section 4 we showed that $L_T$ is an effective regular language, and, in Section 5 we showed that $L_P$ is also an effective regular language. We deduce that $L_T \cap L_P$ is therefore an effective regular language, and that $k$ is computable (since the finiteness and the length of the longest word of a regular language are computable). With a careful analysis of the automata that come into play, we can give an upper bound on $k$. 
**Theorem 4.** \(sd(S)\) is computable, and if \(sd(S) < \infty\) then \(sd(S) < |S|^28|P|^2\), where \(|S|\) is the number of global control states and \(|P|\) the number of processes.

**Proof.** The fact that \(sd(S)\) is computable is explained at the beginning of Section 6. We therefore only prove the claim that, when \(k < \infty\) it holds that \(k < |S|^28|P|^2\). \(k\) is the length of the longest word in \(L_T \cap L_P\). By Theorem 2

\[
L_T = \bigcup R
\]

and by Theorem 3 there is an automaton \(A\) such that \(L(A) = L_P\). So we need to bound the length of the longest word of

\[
L(SR(l, mid, l')) \cap L(A(B, B')) \cap L(A)
\]

assuming that this language is finite, for any \(l, mid, l', B, B'\). This bound is given by the number of states of any automaton that accepts this language (since any longer word would require the automaton to feature a loop, and the language would not be finite). This language is recognized by an automaton that is the product of the automata \(SR(l, mid, fin), A(B, B')\), and \(A\), so its number of states is bounded by

\[
|SR(l, mid, l')| \times |A(B, B')| \times |A|
\]

By definition of \(SR\), \(L_{SR} = L_{S}^2\), so \(|SR| \leq |L_{S}|^2\) (which we can also write \(|S|^2\)). By definition of \(A(B, B')\), \(L_{(B, B')} = B = (2^P \times 2^P)^P\), so \(|A(B, B')| \leq 2^{2|P|^2}\). Finally, by Theorem 3 \(|A| \leq 2^{6|P|^2}\). All together,

\[
k \leq |S|^22^{6|P|^2}
\]

As an immediate consequence of Theorems 1 and 4 we get the following.

**Theorem 5.** The following problem is decidable: given a system \(S\), does there exists a \(k\) such that \(S\) is \(k\)-synchronizable.

### 7 Conclusion

We established that it is possible to determine whether there exists a bound \(k\) such that a given communicating system is \(k\)-synchronizable. For this, we showed how the set of sequences of actions that compose an exchange of arbitrary size can be represented as a regular language, which was possible thanks to the mailbox semantics of communications. We leave for future work to decide whether it would be possible to extend our result to peer-to-peer semantics.
References

1. Abdulla, P.A., Jonsson, B.: Verifying programs with unreliable channels. Inf. Comput. 127(2), 91–101 (1996)
2. Basu, S., Bultan, T.: On deciding synchronizability for asynchronously communicating systems. Theor. Comput. Sci. 656, 60–75 (2016)
3. Bouajjani, A., Enea, C., Ji, K., Qadeer, S.: On the completeness of verifying message passing programs under bounded asynchrony. In: CAV 2018. pp. 372–391 (2018)
4. Brand, D., Zafiropulo, P.: On communicating finite-state machines. J. ACM 30(2), 323–342 (1983)
5. Chaouch-Saad, M., Charron-Bost, B., Merz, S.: A reduction theorem for the verification of round-based distributed algorithms. In: RP 2009. vol. 5797, pp. 93–106. Springer (2009)
6. Chou, C., Gafni, E.: Understanding and verifying distributed algorithms using stratified decomposition. In: Proceedings of the Seventh Annual ACM Symposium on Principles of Distributed Computing, 1988. pp. 44–55. ACM (1988)
7. Di Giusto, C., Laversa, L., Lozes, É.: On the k-synchronizability of systems. In: Goubault-Larrecq, J., König, B. (eds.) FOSSACS 2020, ETAPS 2020. vol. 12077, pp. 157–176. Springer (2020)
8. Elrad, T., Francez, N.: Decomposition of distributed programs into communication-closed layers. Sci. Comput. Program. 2(3), 155–173 (1982)
9. Finkel, A., Lozes, É.: Synchronizability of communicating finite state machines is not decidable. In: ICALP 2017. pp. 122:1–122:14 (2017)
10. Genest, B., Kuske, D., Muscholl, A.: On communicating automata with bounded channels. Fundam. Inform. 80(1-3), 147–167 (2007)
11. von Gleissenthal, K., Kici, R.G., Bakst, A., Stefan, D., Jhala, R.: Pretend synchrony: synchronous verification of asynchronous distributed programs. PACMPL 3(POPL), 59:1–59:30 (2019)
12. Kragl, B., Qadeer, S., Henzinger, T.A.: Synchronizing the asynchronous. In: CONCUR 2018. vol. 118, pp. 21:1–21:17 (2018)
13. La Torre, S., Madhusudan, P., Parlato, G.: Context-bounded analysis of concurrent queue systems. In: TACAS 2008, ETAPS 2008. pp. 299–314 (2008)
14. Lipton, R.J.: Reduction: A method of proving properties of parallel programs. Commun. ACM 18(12), 717–721 (1975)