On the decidability of synchronizability for mailbox communicating automata

—

Internship Report

Laetitia Laversa
Master 2 RIF
I3S Laboratory - Teams MDSC & COMRED
Cinzia Di Giusto, Etienne Lozes

September 1, 2018

Abstract

Distributed systems are increasingly common and it is therefore important to ensure that they are error-free. We are interested in how they exchange information and more particularly in the property of synchronizability. A system is synchronizable if its behaviour when considering asynchronous communication is the same as in the case of synchronous communication. Systems are modelled as networks of communicating automata and we focus on peer-to-peer and mailbox communications. By reducing the Post correspondance problem to synchronizability, we give an alternative proof that synchronizability is undecidable for peer-to-peer systems and prove that it is undecidable for mailbox systems with finite states.

Contents

1 Introduction

We observe an increasing use of distributed systems in different domains, such as embedded systems, multiprocessor hardware architecture, communications protocols, web applications, etc. It is necessary to ensure proper functioning and absence of errors, especially for the most critical which could cause damage in the event of malfunctioning. This requires an understanding of how these systems work and what the sources of errors may be. A distributed system is composed of many components, named *peers*. Peers exchange messages to coordinate and reach a common objective. Usually, such communications are asynchronous. This entails that a message sent can be stored for an indefinite time before it is read. This asynchronous communication is error prone, because it is difficult to implement and to execute in a reproducible way. Whereas synchronous communication, where a message is sent and received simultaneously, can easily be executed in a reproducible way. For the same underlying system, the number of possible behaviours is, in general, greater with asynchronous communication than with synchronous communication. Since synchronous communication is easier, we are interested in identifying all those systems whose synchronous behaviour is the same as the asynchronous one. In other words all those systems where the type of communication does not influence the overall behaviour. This property is called synchronizability.

"Real" distributed systems may be rather complex, combining advanced hardware features as well as involved management algorithms. Our focus is on communication, therefore we will abstract away from any implementation details. We choose to work with networks of communicating automata [**?**, **?**], that are a standard model for distributed systems and they allow to reason about communication protocols [**?**, **?**]. Communicating automata, intuitively, are finite state automata where arc labels represent channels and allow to exchange messages in a network of automata or peers. Communication between peers can take different forms, here we only consider communication via FIFO buffers, either used in a peer-to-peer fashion (one buffer per pair of machines), or operating as a mailbox (one buffer per receiver). As in [**?**, **?**, **?**, **?**], we assume that these buffers are unbounded because in general, we cannot compute the number of messages that the buffer is likely to contain (the problem is undecidable).

Most problems on synchronous communicating automata are decidable, as for example the problem of reachability, while conversely, the majority of problems on asynchronous communicating automata are undecidable. The intrinsic complexity of asynchronous distributed systems and the simplicity of synchronous ones lead us to the notion of synchronizability: a system is synchronizable if any asynchronous behaviour can be mimicked by a synchronous one that contains the same send actions in the same order. This notion has already been studied, particularly in [**?**] by Basu and Bultan,

where the authors claim the decidability of synchronizability. Nonetheless, in [?] Finkel and Lozes show that previous claim is false. They prove the undecidability of the synchronizability for peer-to-peer systems and give a counter-example that the synchronizability for mailbox systems is not decidable. We will comment more on their work in Section ??. Summing up, the synchronizability for peer-to-peer systems is undecidable but the problem for mailbox systems is still open.

The objective of this report, is, thus, to assess the decidability of the synchronizability for mailbox systems. We prove the undecidability of the synchronizability for mailbox systems, with some additional constraints, resorting to the Post correspondence problem which is an undecidable problem. Moreover, the study of the decidability of synchronizability led us to an alternative proof of undecidability, different from the existing one, for peer-to-peer systems. Networks of communicating automata can be big and complex, so it is normal to use tools to analyse them. We use a tool and adapt it to meet our needs (STABC, discussed in Section ??) to check whether a system is not synchronizable.

Overview. In Section ??, we define networks of communicating automata, synchronizability, and the Post correspondence problem. In Section ??, we introduce the tools: CADP and STABC and describe their uses and the changes we have made to STABC. Sections ?? and ?? concern the decidability of the synchronizability problem for peer-to-peer systems and mailbox systems, respectively. In Section ??, we comment on previous works discussing the notion of synchronizability. Finally, Section ?? concludes this report and opens up some perspectives.

2 Preliminaries

2.1 Communicating automata

A communicating automaton is a finite state machine that performs only two basic operations to move from a state to another: either sending or receiving messages from other communicating automata. A *network* of communicating automata, or simply network, is a parallel composition of a finite set P of n peers. In a network, communicating automata can communicate with each other. We consider a finite set of messages M. Each message in M consists of a sender, a receiver, and some finite information. We denote $a^{p \to q} \in M$ the message sent from peer p to peer q with information a. For all messages $a^{p \to q}$, $a'^{p' \to q'} \in M$, we assume the following:

1. $p \neq q$, i.e., a peer can not send a message to itself,
2. if $a = a'$ then $p = p'$ and $q = q'$, i.e., each information can be sent by a unique sender and can be received by a unique receiver.

Thanks to the second assumption, we will sometimes omit the peers and simply write a for a message $a^{p\to q}$ in M. An action is the sending or the reception of a message, we denote $!m$ the sending and $?m$ the reception of $m \in M$. Therefore, we define $Act_p = \{!a^{p\to q} \mid a^{p\to q} \in M, q \in P\} \cup \{?a^{q\to p} \mid a^{q\to p} \in M, q \in P\}$ the set of actions for peer p.

Definition 1 (*Network of communicating automata*)**.**
Let $\mathcal{N} = ((\mathcal{A}_p)_{p\in P}, M, F)$ be a network of communicating automata, where:

- For each $p \in P$, $\mathcal{A}_p = (S_p, s_0^p, M, \to_p)$ is a communicating automaton with
  - a finite set S_p of states
  - $s_0^p \in S_p$ a distinguished initial state
  - $\to_p \subseteq S_p \times Act_p \times S_p$ a transition relation;
- $F \subseteq \prod_{p\in P} S_p$ is the set of global final states, where $\prod$ denote the cartesian product.

Example 2.1 (Network)**.** *Consider network* $\mathcal{N} = ((\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3), M, F)$ *depicted in Figure* **??**. *Its initial state is* (s_0^1, s_0^2, s_0^3)*, the set of global final states is* $F = \{(s_1^1, s_1^2, s_3^3)\}$*, and finally the set of messages is* $M = \{a^{1\to 3}; b^{2\to 3}\}$.

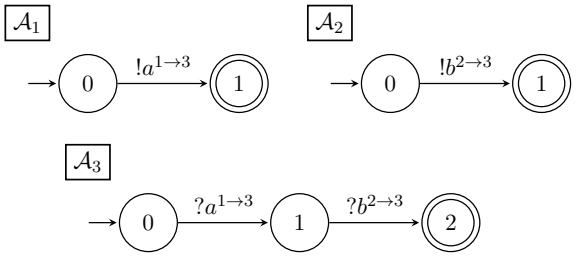


Figure 2.1: Network $\mathcal{N}$

Remark 1. *Notice that state* s_i^p *is depicted by a circled* i *in the figure. Final states are depicted with a double circle.*

Different communication types can be associated to the same network. We call a *system* a network together with a communication type. A system can communicate synchronously or asynchronously. In a synchronous communication, each message sent is immediately received. A message can not be sent if it can not be received. In an asynchronous communication instead, messages are stored in FIFO (First In First Out) buffers, which can be bounded or unbounded. If a buffer is of size k, we can talk about a k-bounded buffer. A FIFO buffer is an ordered data structure where data are stored in a queue. When an element is added, it is stored at the tail of the queue, and when a element is removed, it is taken from its head. Several models of systems are possible.

- Synchronous (denoted 0): there is no buffer in the system, messages are immediately received when there are sent.
- Peer-to-peer (denoted 1 - 1): there is a buffer for each pair of peers, where one element of the pair is the sender and the other is the receiver. A network with a peer-to-peer type of communication composed of n peers contains $n \times (n-1)$ buffers.
- Mailbox (denoted * - 1): there are as many buffers as peers, each peer receives all its messages in a unique buffer, no matter the sender. A network with a mailbox type of communication composed of n peers contains n buffers.

We use *configurations* to describe the state of a system and its buffers.

Definition 2 (*Configuration*)**.** Let $\mathcal{N} = ((\mathcal{A}_p)_{p\in P}, M, F)$ be a network. A 0 configuration (respectively a 1 - 1 configuration, or a * - 1 configuration) is a tuple $C = ((s^p)_{p\in P}, B)$ such that :

- s^p is a state of automaton $\mathcal{A}_p$, for all $p \in P$
- B is the content of all buffers:
  - an empty tuple for a 0 configuration,
  - a tuple $(b_{12}, \ldots, b_{n(n-1)})$ for a 1 - 1 configuration and
  - a tuple $(b_1, \ldots, b_n)$ for a * - 1 configuration,

  where each buffer content $b_i \in M^*$ is a finite sequence of messages.

We write ε to denote an empty buffer, and $B^\emptyset$ to denote that all buffers are empty. $C_0 = ((s_0^p)_{p\in P}, B^\emptyset)$ is the *initial configuration*. We write $B\{b_i/b\}$ the tuple of buffers B where buffer b_i is substituted by b.

Several systems can be defined according to the type of communication.

Definition 3 (*Synchronous System*)**.** Let $\mathcal{N} = ((\mathcal{A}_p)_{p\in P}, M, F)$ be a network. The synchronous system $\mathcal{N}_0$ associated with $\mathcal{N}$ is the least binary relation $\underset{0}{\longrightarrow}$ over 0 configurations such that

$$(0 \text{ SEND}) \quad \frac{s^p \xrightarrow{!a^{p\to q}}_p s'^p \qquad s^q \xrightarrow{?a^{p\to q}}_q s'^q}{((s^1, \ldots, s^p, \ldots, s^q, \ldots, s^n), B^\emptyset) \underset{0}{\xrightarrow{!a^{p\to q}}} ((s^1, \ldots, s'^p, \ldots, s'^q, \ldots, s^n), B^\emptyset)}$$

In a synchronous system, there is no buffer. A message can be sent if a transition $!a^{p\to q}$ exists in the sender automaton p and the corresponding reception $?a^{p\to q}$ exists in the receiver automaton q. As a result, the communication takes places and both automata p and q change their state.

In order to simplify the definitions of traces and language (given in what follows), and without loss of generality, we choose to label the transition with the sending message $!a^{p\to q}$.

Definition 4 (*Peer-to-peer System*). Let $\mathcal{N} = ((\mathcal{A}_p)_{p\in P}, M, F)$ be a network, and let $k \in \mathbb{N}^+ \cup \{\infty\}$ be a fixed buffer bound. The peer-to-peer system $\mathcal{N}_{1-1^k}$ associated with $\mathcal{N}$ and k is the least binary relation $\xrightarrow[1-1^k]{}$ over $1-1$ configurations such that:

- for each configuration $C = ((s^p)_{p\in P}, B)$, $B = (b_{pq})_{p,q\in P}$ where $b_{pq} \in M^*$ is of length $|b_{pq}| \leq k$.
- $\xrightarrow[1-1^k]{}$ is the least transition induced by:

$$\text{(1-1 SEND)}\ \frac{s^p \xrightarrow{!a^{p\to q}}_p s'^p \qquad |b_{pq}| < k}{((s^1,\ldots,s^p,\ldots,s^n),B) \xrightarrow[1-1^k]{!a^{p\to q}} ((s^1,\ldots,s'^p,\ldots,s^n), B\{b_{pq}/b_{pq}\cdot a\})}$$

$$\text{(1-1 REC)}\ \frac{s^q \xrightarrow{?a^{p\to q}}_q s'^q \qquad b_{pq} = a\cdot b'_{pq}}{((s^1,\ldots,s^q,\ldots,s^n),B) \xrightarrow[1-1^k]{?a^{p\to q}} ((s^1,\ldots,s'^q,\ldots,s^n), B\{b_{pq}/b'_{pq}\})}$$

Definition 5 (*Mailbox System*). Let $\mathcal{N} = ((\mathcal{A}_p)_{p\in P}, M, F)$ be a network and let $k \in \mathbb{N}^+ \cup \{\infty\}$ be a fixed buffer bound. The peer-to-peer system $\mathcal{N}_{*-1^k}$ associated with $\mathcal{N}$ and k is the least binary relation $\xrightarrow[*-1^k]{}$ over $*-1$ configurations such that:

- for each configuration $C = ((s^p)_{p\in P}, B)$, $B = (b_p)_{p\in P}$ where b_p is of length $|b_p| \leq k$.
- $\xrightarrow[*-1^k]{}$ is the least transition such that :

$$\text{(*-1 SEND)}\ \frac{s^p \xrightarrow{!a^{p\to q}}_p s'^p \qquad |b_q| < k}{((s^1,\ldots,s^p,\ldots,s^n),B) \xrightarrow[*-1^k]{!a^{p\to q}} ((s^1,\ldots,s'^p,\ldots,s^n), B\{b_q/b_q\cdot a\})}$$

$$\text{(*-1 REC)}\ \frac{s^q \xrightarrow{?a^{p\to q}}_q s'^q \qquad b_q = a\cdot b'_q}{((s^1,\ldots,s^q,\ldots,s^n),B) \xrightarrow[*-1^k]{?a^{p\to q}} ((s^1,\ldots,s'^q,\ldots,s^n), B\{b_q/b'_q\})}$$

In peer-to-peer and mailbox systems, a sending can be done if the transition $!a^{p\to q}$ exists in the state of sender automaton p and if the target buffer is not full. The message is put on the tail of the buffer. A reception of a message can be done if transition $?a^{p\to q}$ exists in receiver automaton q and if this message is at the head of the buffer. As a result, the message is removed from the buffer. Notice that, in mailbox systems, a same buffer can contain messages from different senders, which, as we will see in the following, imposes more constraints.

We describe the behaviour of a system with *runs*. A run is a sequence of transitions starting from an initial configuration C_0.

Example 2.2 (Buffers layout and run)**.** *We use network $\mathcal{N}$ of Example* **??**. *The topology of the network is described in Figure* **??**.

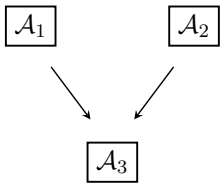


Figure 2.2: Topology of Network $\mathcal{N}$

Several buffer layouts are possible according to the type of communication.

- *(Synchronous communication) In system $\mathcal{N}_0$, there are no buffers. Messages are directly read by the receiving peer.*
- *(Peer-to-peer communication) In system $\mathcal{N}_{1-1^k}$, there is one buffer of size k per pair of peers. Figure* **??** *represents the system where the buffers that are represented are the ones that are used for communications, namely b_{13} and b_{23}.*

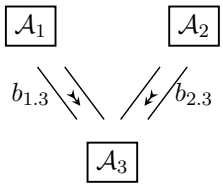


Figure 2.3: Buffer layout of system $\mathcal{N}_{1-1}$

- *(Mailbox communication) In system $\mathcal{N}_{*-1^k}$, there is one buffer of size k per peers. The Figure* **??** *represents the system with depicted the only used buffer b_3.*

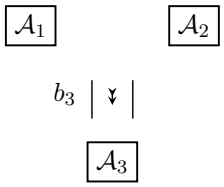


Figure 2.4: Buffer layout of $\mathcal{N}_{*-1}$

Clearly each type of communication gives rise to different runs. We give an example of run for each type of communication.

- *(Synchronous communication) The only possible run is:*

$$((s_0^1, s_0^2, s_0^3), \emptyset) \xrightarrow[0]{!a^{1\to 3}} ((s_1^1, s_0^2, s_1^3), \emptyset) \xrightarrow[0]{!b^{2\to 3}} ((s_1^1, s_1^2, s_2^3), \emptyset)$$

- *(Peer-to-peer communication) A possible run is:*

$$((s_0^1, s_0^2, s_0^3), \{b_{13} = \varepsilon, b_{23} = \varepsilon\}) \xrightarrow[1-1^\infty]{!b^{2\to 3}} ((s_0^1, s_1^2, s_0^3), \{b_{13} = \varepsilon, b_{23} = b\})$$

$$((s_0^1, s_1^2, s_0^3), \{b_{13} = \varepsilon, b_{23} = b\}) \xrightarrow[1-1^\infty]{!a^{1\to 3}} ((s_1^1, s_1^2, s_0^3), \{b_{13} = a, b_{23} = b\})$$

$$((s_1^1, s_1^2, s_0^3), \{b_{13} = a, b_{23} = b\}) \xrightarrow[1-1^\infty]{?a^{1\to 3}} ((s_1^1, s_1^2, s_1^3), \{b_{13} = \varepsilon, b_{23} = b\})$$

$$((s_1^1, s_1^2, s_1^3), \{b_{13} = \varepsilon, b_{23} = b\}) \xrightarrow[1-1^\infty]{?b^{2\to 3}} ((s_1^1, s_1^2, s_2^3), \{b_{13} = \varepsilon, b_{23} = \varepsilon\})$$

- *(Mailbox communication) A possible run is:*

$$((s_0^1, s_0^2, s_0^3), \{b_3 = \varepsilon\}) \xrightarrow[*-1^\infty]{!b^{2\to 3}} ((s_0^1, s_1^2, s_0^3), \{b_3 = b\})$$

$$((s_0^1, s_1^2, s_0^3), \{b_3 = b\}) \xrightarrow[*-1^\infty]{!a^{1\to 3}} ((s_1^1, s_1^2, s_0^3), \{b_3 = ba\})$$

We remark that with the same sendings, we can not reach the same global state with system $\mathcal{N}_{1-1^\infty}$ and $\mathcal{N}_{*-1^\infty}$. Indeed, as said earlier, mailbox communication imposes more constraints. As we have one buffer for $\mathcal{A}_3$, no matter the sender, and this buffer is a FIFO (First In First Out), we can not receive a before b while b was sent before a.

Let $com \in \{0; 1-1^k; *-1^k\}$ be the type of communication with k the size of buffers, we define $\xrightarrow[com]{}{}^*$ as the transitive reflexive closure of $\xrightarrow[com]{}$. We denote with $\xrightarrow[com]{?^*}$ a sequence of receptions only and with $\overset{!a^{p\to q}}{\underset{com}{\Longrightarrow}}$ a sequence of the kind $\xrightarrow[com]{!a^{p\to q}} \xrightarrow[com]{?^*}$, that corresponds to one send followed by zero or more receptions.

In order to study the behaviour of systems, we define the set of traces and the associated language. A trace t is a sequence of actions.

Definition 6 (*Trace*)**.** Let $\mathcal{N} = ((\mathcal{A}_p)_{p\in P}, M, F)$ be a network and $com \in \{0; 1-1^k; *-1^k\}$ be the type of communication with k the size of the buffers. $T(\mathcal{N}_{com})$ is the set of traces defined by

$$T(\mathcal{N}_{com}) = \{act_1 \cdot \ \ldots \ \cdot act_e \mid C_0 \xrightarrow[com]{act_1} C_1 \xrightarrow[com]{act_2} \ldots \xrightarrow[com]{act_e} C_e\}$$

where C_0 is the initial configuration.

Similarly, a send trace t^s is a sequence of sendings.

Definition 7 (*Send trace*). Let $\mathcal{N} = ((\mathcal{A}_p)_{p\in P}, M, F)$ be a network and $com \in \{0; 1-1^k; *-1^k\}$ be the type of communication with k the size of buffers. $T^s(\mathcal{N}_{com})$ is the set of send traces such that

$$T^s(\mathcal{N}_{com}) = \{act_1 \cdot \ \ldots \ \cdot act_e \mid C_0 \overset{act_1}{\underset{com}{\Longrightarrow}} C_1 \overset{act_2}{\underset{com}{\Longrightarrow}} \ldots \overset{act_e}{\underset{com}{\Longrightarrow}} C_e\}$$

where C_0 is the initial configuration.

For a synchronous system, the set of traces is the same as the set of send traces, because of the labelling of the transitions.

Property 1. *For all network $\mathcal{N}$, we have $T(\mathcal{N}_0) = T^s(\mathcal{N}_0)$.*

A final send trace t^f is a sequence of sendings where the last configurations reached contains a final global state.

Definition 8 (*Final send trace*). Let $\mathcal{N} = ((\mathcal{A}_p)_{p\in P}, M, F)$ be a network and $com \in \{0; 1-1^k; *-1^k\}$ be the type of communication with k the size of the buffers. $T^f(\mathcal{N}_{com})$ is the set of final send traces such that

$$T^f(\mathcal{N}_{com}) = \{act_1 \cdot \ \ldots \ \cdot act_e \mid C_0 \overset{act_1}{\underset{com}{\Longrightarrow}} C_1 \overset{act_2}{\underset{com}{\Longrightarrow}} \ldots \overset{act_e}{\underset{com}{\Longrightarrow}} C_e\}$$

where C_0 is the initial configuration and $C_e = (S_g, B)$ with $S_g \in F$.

Remark 2. *In Definition ??, we decide not to take into consideration the content of buffers, differently to others papers, like [?], where the authors study stable configurations, i.e., configurations where buffers are empty.*

The language is the set of all possible final send traces of the system.

Definition 9 (*Language*). Let $com \in \{0; 1-1^k; *-1^k\}$ be the type of communication where k is the size of buffers and $\mathcal{N}_{com} = (\mathcal{N}, \underset{com}{\longrightarrow})$ a system. Its language is $\mathcal{L}(\mathcal{N}_{com}) = T^f(\mathcal{N}_{com})$.

Example 2.3 (Traces and language). *We use network $\mathcal{N}$ of Example ??. The set $T(\mathcal{N}_{*-1^2})$ of system $\mathcal{N}_{*-1^2}$ is composed of traces:*

- $!a^{1\to3} \cdot !b^{2\to3} \cdot ?a^{1\to3} \cdot ?b^{2\to3}$
- $!a^{1\to3} \cdot ?a^{1\to3} \cdot !b^{2\to3} \cdot ?b^{2\to3}$
- $!b^{2\to3} \cdot !a^{1\to3}$

and of all prefixes of these traces.
The set of send traces of this system is:

$$T^s(\mathcal{N}_{*-1^2}) = \{!a^{1\to3};\ !b^{2\to3};\ !a^{1\to3} \cdot !b^{2\to3};\ !b^{2\to3} \cdot !a^{1\to3}\}$$

The set of final send traces of this system is:

$$T^f(\mathcal{N}_{*-1^2}) = \{!a^{1\to3} \cdot !b^{2\to3}\}$$

Indeed, the send trace $!b^{2\to3} \cdot !a^{1\to3}$ *does not allow to reach a final global state, because after these sendings, the system is in configuration*

$$C = ((s_1^1, s_1^2, s_0^3), \{b_3 = ba\})$$

where $\mathcal{A}_3$ *cannot reach state* s_2^3*. Thus, the system cannot reach its global final state, so this send trace is not a final send trace.*
The language of system $\mathcal{N}_{*-1^2}$ *is* $\mathcal{L}(\mathcal{N}_{*-1^2}) = \{!a^{1\to3} \cdot !b^{2\to3}\}$.

Remark 3. *In what follows, we will sometimes consider networks* $\mathcal{N}$ *that do not have final states, which amounts to having all final states. We notice that in these cases,* $\mathcal{L}(\mathcal{N}_{com}) = T^f(\mathcal{N}_{com}) = T^s(\mathcal{N}_{com})$*, i.e., their language corresponds to their set of send traces.*

2.2 Synchronizability Problem

A system is synchronizable if its asynchronous behaviour can be related to its synchronous one. Thus, an asynchronous system is synchronizable if its language is the same as the one obtained from the synchronous system. More precisely:

Definition 10 (*Synchronizability*)**.** Let $type \in \{1-1; *-1\}$ and $\mathcal{N}$ be a network. A system $\mathcal{N}_{type^\infty}$ is synchronizable if and only if

$$\mathcal{L}(\mathcal{N}_{type^\infty}) = \mathcal{L}(\mathcal{N}_0)$$

The Synchronizability Problem is the decision problem of determining whether a given system is synchronizable or not.

When a system is not synchronizable, we can be interested in its stability. A system is k-stable if increasing the size of buffers does not change its language. More precisely:

Definition 11 (*Stability*)**.** Let $type \in \{1-1; *-1\}$ and $\mathcal{N}$ be a network. A system is k-stable if and only if

$$\exists k \in \mathbb{N}^+ \text{ such that } \mathcal{L}(\mathcal{N}_{type^k}) = \mathcal{L}(\mathcal{N}_{type^\infty})$$

2.3 Post Correspondence Problem

We will use the Post Correspondence Problem (PCP) to prove that the Synchronizability Problem is undecidable.

Definition 12 (*Word*)**.** Let Σ an alphabet, a word $w = a_1 a_2 \dots a_n \in \Sigma^*$ is a finite sequence of symbols, such that $a_i \in \Sigma$, $\forall$ i $\in [1, n]$.

Let $w_a \in \Sigma^*$ and $w_b \in \Sigma^*$, and $m, n \in \mathbb{N}$, such that $w_a = w_{a,1} w_{a,2} \dots w_{a,m}$ and $w_b = w_{b,1} w_{b,2} \dots w_{b,n}$. The concatenation of w_a and w_b, written $w_a \cdot w_b$, corresponds to the word $w_{a,1} w_{a,2} \dots w_{a,m} w_{b,1} w_{b,2} \dots w_{b,n}$. We denote $|w|$ the size of the word w.

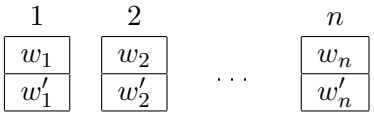
Definition 13 (*Post Correspondence Problem*)**.** Let Σ be an alphabet with at least two symbols. An instance (W,W') of PCP consists of two finite ordered lists of non-null words of the same length

$$\text{W} = w_1, w_2, \ldots, w_n \text{ and W'} = w'_1, w'_2, \ldots, w'_n$$

such that $w_i, w'_i \in \Sigma^*$ for all indices $i \in [1, n]$. A solution of this instance is a finite sequence of indices $Sol = (i_1, i_2, \ldots i_m)$ with $m \geq 1$ and $\forall$ j $\in [1, m]$, $i_j \in [1, n]$, such that:

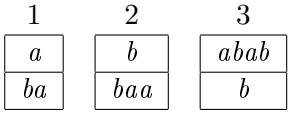
$$w_{i_1} \cdot w_{i_2} \cdot \cdots \cdot w_{i_m} = w'_{i_1} \cdot w'_{i_2} \cdot \cdots \cdot w'_{i_m}$$

We can represent the input as a set of n blocks:

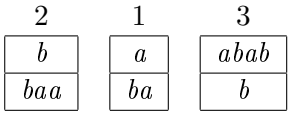


The PCP is an undecidable decision problem [**?**, **?**].

Example 2.4 (Instance of PCP)**.** *Let* $\Sigma = \{a, b\}$.
Input: $W = a, b, abab$ *and* $W' = ba, baa, b$*, i.e.,*



We have a solution for this instance with $S = (2, 1, 3)$*:*



where $w_2 \cdot w_1 \cdot w_3 = w'_2 \cdot w'_1 \cdot w'_3 = baabab$.

3 Tools

We use some tools allowing to study networks of communicating automata and verify properties such as synchronizability and stability. Notice that the notion of global final state, and so the notion of final send traces are absent in these tools. So, only in this section, the language of a system is defined by the set of send traces.

3.1 CADP

CADP[1] is a generic toolbox for the design of synchronous concurrent systems [?], so it allows us to model and study our networks of communicating automata. It based on concepts from CCS [?]. It takes the specification of the system in the LOTOS language [?].

Here, we use it to generate and visualize the set of send traces in the form of state graphs, but also to compare these state graphs according to an equivalence notion. It is directly called by STABC, and we seldom used CADP tools directly.

3.2 STABC

STABC[2] is a tool based on CADP, coded in Python. It takes as input a network and determines if this network is synchronizable or k-stable by calling CADP. CADP can manipulate only synchronizable systems, however STABC generates LOTOS systems that encodes the bounded buffers. It considers only mailbox systems.

It determines these properties according to an equivalence notion chosen between *branching*, *strong* and *trace* (defined in [?]). It creates state graphs of synchronous and asynchronous systems, considering either all messages or only send messages, reduces it with the chosen equivalence notion before to compare it, always according to this notion.

STABC claims synchronizability if the state graph obtained for the synchronous system is equal to that obtained for the asynchronous system with 1-bounded buffers (property stated in [?]). To determine stability of a system with k-bounded buffers, STABC compares state graph of this system with the one of the system based on the same network with buffers of size $k+1$ (property stated in [?]).

We can write the pseudo-code of the algorithm used to determine synchronizability and stability to understand it (Algorithm **??**). We give to STABC a network $\mathcal{N}$, an equivalence notion eq and an integer k_{max}. We write $\sim_{eq}$ to denote the equivalence between two systems according to the equivalence notion eq. We write *compute* to indicate the building of state graph of a system.

Note in this algorithm that STABC is based on properties that, as we will see below, are not trustable. We actually modified STABC to detect *non*-synchronizability.

3.3 Modifications and uses

Theorem 2 in [?] claims that if the language of an asynchronous system with 1-bounded buffers is the same than the one of the synchronous system based

[1] https://cadp.inria.fr/

[2] http://convecs.inria.fr/people/Gwen.Salaun/Tools/index.html

Algorithm 1: Pseudo-code of *checkBruteForce*()

```
compute N_0
compute N_{*-1^1}
if N_0 ~_eq N_{*-1^1} then
    return "synchronizable"
else
    k = 1
    while k ≤ k_max do
        compute N_{*-1^{k+1}}
        if N_{*-1^k} ~_eq N_{*-1^{k+1}} then
            return "k-stable"
        else
            k ← k + 1
        end
    end
end
```

on the same network, then this system is synchronizable. More formally, for a network $\mathcal{N}$, if $\mathcal{L}(\mathcal{N}_0) = \mathcal{L}(\mathcal{N}_{*-1^1})$ then $\mathcal{L}(\mathcal{N}_0) = \mathcal{L}(\mathcal{N}_{*-1^\infty})$. This theorem is used by STABC to determine the synchronizability.

A counter-example (Figure **??**) in [**?**] demonstrates that this theorem is false for mailbox systems.

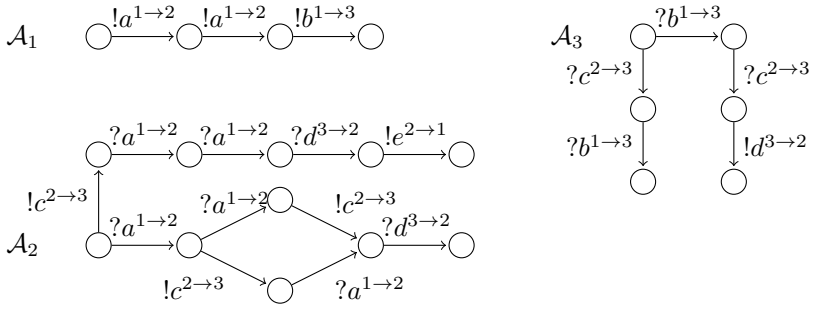


Figure 3.1: Counter-example 1 (from [**?**])

The message $e^{2\to1}$ can be sent only if buffers are at least of size 2. Therefore, this network contradicts the theorem. Indeed, we have:

$$\mathcal{L}(\mathcal{N}_0) = \mathcal{L}(\mathcal{N}_{*-1^1}) \neq \mathcal{L}(\mathcal{N}_{*-1^2})$$

We want to confirm it using STABC. However, as mentioned earlier,

Algorithm 2: Pseudo-code of *checkMyStrat*()

```
compute N_0
compute N_{*-1^1}
if N_0 ~_weak N_{*-1^1} then
    print "L(N_0) = L(N_{*-1^1})"
else
    print "not synchronizable"
end
k = 1
while k ≤ k_max do
    compute N_{*-1^{k+1}}
    if N_{*-1^k} ~_weak N_{*-1^{k+1}} then
        print "L(N_{*-1^k}) = L(N_{*-1^{k+1}})"
    else
        print "not k-stable"
    end
    k ← k + 1
end
```

STABC uses this same theorem to determine synchronizability and compares only the state graphs of the synchronous system and the asynchronous system with 1-bounded buffers. Therefore, we need to modify this tool to verify the validity of this counter-example, i.e., compares these state graphs but also the one of the asynchronous system with 2-bounded buffers. Moreover, the notion of equivalence that we want to use (which corresponds to the notion of send trace) is the *weak* equivalence (defined in [**?**]). STABC is not able to use it, but we can modify it to use this notion which is already present in CADP.

Therefore, we defined a new function, *checkMyStrat*(), based on a function, *checkBruteForce*(), already implemented in STABC. *checkMyStrat*() creates state graphs of the synchronous system and compares it with the one of the asynchronous system with 1-bounded buffers. If they are different, the system is not synchronizable. Then, the asynchronous system with 1-bounded buffers is compared with the asynchronous system with 2-bounded buffers, and so on until it finds a difference or it reach a bound passed as a parameter of the function. Thus, with this new function, STABC can not say if a system is synchronizable or k-stable (it would be necessary to compare it to the system with unbounded buffers, which CADP cannot handle) but can say if it is not. Algorithm **??** gives the pseudo-code for *checkMyStrat*() function.

We have tested our function on the network of Figure **??**. As expected, our function indicates that the set of send traces of the synchronous system is equal to that of the asynchronous system with a 1-bounded buffer, and

that the set of send traces of the asynchronous system with a 1-bounded buffer is different from that of the asynchronous system with a 2-bounded buffer.

This theorem being false, Theorem 1 in [**?**], which claims a condition to ensure stability (similar to the one of synchronizability), seems to be false as well. This theorem claims that, if an asynchronous system with k-bounded buffers has the same language as the same asynchronous system with buffers of size $k+1$, then this system is stable. More formally, for a network $\mathcal{N}$, if $\mathcal{L}(\mathcal{N}_{*-1^k}) = \mathcal{L}(\mathcal{N}_{*-1^{k+1}})$ then $\mathcal{L}(\mathcal{N}_{*-1^k}) = \mathcal{L}(\mathcal{N}_{*-1^\infty})$.

We want to confirm that this theorem is false, building a new counter-example, depicted in Figure **??**, inspired by the previous one, and testing it with the modified STABC.

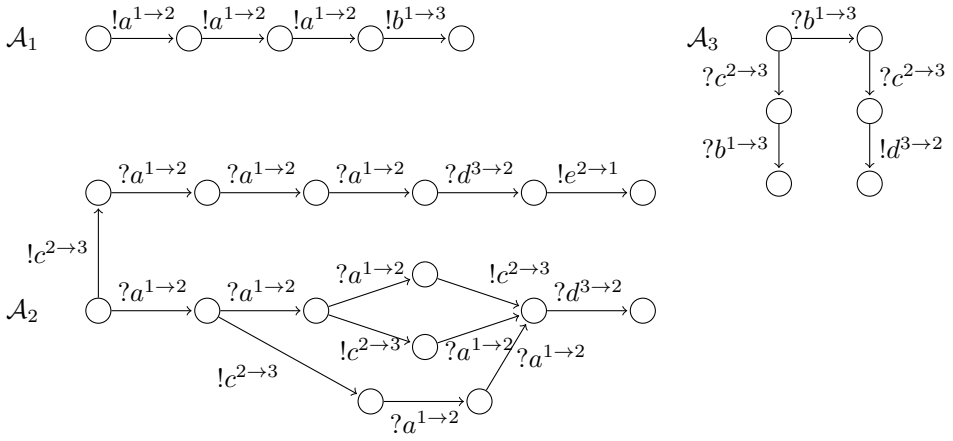


Figure 3.2: Counter-example 2

As expected, message e can be sent only with buffers of size 3 at least, and so we have:

$$\mathcal{L}(\mathcal{N}_0) \neq \mathcal{L}(\mathcal{N}_{*-1^1}) = \mathcal{L}(\mathcal{N}_{*-1^2}) \neq \mathcal{L}(\mathcal{N}_{*-1^3})$$

that contradicts this theorem.

Remark 4. *In the journal version of this paper [**?**], published in 2018, the authors define differently the notion of Asynchronous composition with FIFO buffers. Theorem 1 with the new definition appears to be true.*

The changes to STABC allowed us to detect some errors in the PCP encodings that we tried while working on the conjectures of the following sections.

4 Synchronizability Problem for Peer-to-peer Systems

In this section, we show the undecidability of synchronizability for systems with peer-to-peer communication. The proof is a reduction from PCP. We encode an instance of PCP into a network of communicating automata. We denote $\mathcal{N} = [\![\mathcal{W}, \mathcal{W}']\!]^{p2p}$ the encoding of instance (W, W'). We prove that a solution to this instance exists if and only if

$$\mathcal{L}(\mathcal{N}_0) \neq \mathcal{L}(\mathcal{N}_{1-1^\infty})$$

4.1 Peer-to-peer Encoding of Post Correspondence Problem

The encoding of an instance (W,W') of PCP is the parallel composition of four automata : $\mathcal{A}_W$, $\mathcal{A}_{W'}$, $\mathcal{A}_L$ and $\mathcal{A}_I$. The topology of the network is depicted in Figure **??**.

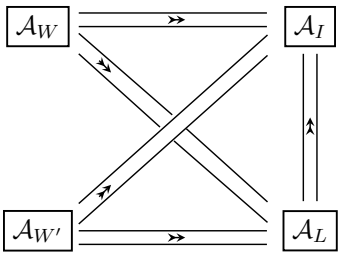


Figure 4.1: Topology of $\mathcal{N}$

$\mathcal{A}_W$ and $\mathcal{A}_{W'}$ send a sequence of indices to $\mathcal{A}_I$ and corresponding words to $\mathcal{A}_L$. $\mathcal{A}_I$ checks whether indices sent by $\mathcal{A}_W$ and $\mathcal{A}_{W'}$ are the same, and $\mathcal{A}_L$ does the same with letters.

Message *ok* allows to know if both comparisons, letters and indices, have succeeded. If the comparison of letters succeed till the end of both sequences, then $\mathcal{A}_L$ can send a message *ok* to $\mathcal{A}_I$. Whenever $\mathcal{A}_I$ fails, it goes to a state where it can receive message *ok*. But, if comparisons of indices succeed till the end, $\mathcal{A}_I$ finish in a state where it can not receive it.

The global final state of this encoding is reached when $\mathcal{A}_W$ and $\mathcal{A}_{W'}$ have sent messages indicating the end of their words, when $\mathcal{A}_I$ and $\mathcal{A}_L$ succeeded all their comparisons, and when $\mathcal{A}_L$ have also sent message *ok*.

Notice that with synchronous communication, a message can not be sent if it can not be received. Thus, if the instance of PCP has a solution, the comparison of letters and the one of indices have to succeed. But within synchronous communication, message *ok* can not be sent. Whereas within asynchronous communication, message *ok* will be sent and the final global state will be reached.

More formally, we define the encoding as follows:

Definition 14 (*Encoding of PCP in peer-to-peer system*)**.** Let (W,W') be a PCP instance over Σ, where $W = w_1, \ldots, w_n$ and $W' = w'_1, \ldots, w'_n$. The encoding of (W,W') is $[\![\mathcal{W}, \mathcal{W}']\!]^{p2p} = ((\mathcal{A}_p)_{p \in P}, M, F)$ where:

- $P = (W, W', L, I)$;
- The set of messages is

$$\begin{aligned} M = {} & \{i^{W \to I} \mid i \in [1,n]\} \cup \{\$^{W \to I}\} \cup \{\alpha^{W \to L} \mid \alpha \in \Sigma\} \cup \{end^{W \to L}\} \cup \\ & \{i'^{W' \to I} \mid i \in [1,n]\} \cup \{\$'^{W' \to I}\} \cup \{\alpha'^{W' \to L} \mid \alpha \in \Sigma\} \cup \\ & \{end'^{W' \to L}\} \cup \{ok^{L \to I}\}; \end{aligned}$$

- $\mathcal{A}_W = (S_W, s_0^W, M, \longrightarrow_W)$ where:

$$\begin{aligned} & S_W = \{q_0, q_f, q_\$, q_e\} \cup \{q_{i,j} \mid i \in [1,n], j \in [0, |w_i| - 1]\ \} \\ & s_0^W = q_0 \\ & \longrightarrow_W = \{q_0 \xrightarrow{!i^{W \to I}} q_{i,0} \mid i \in [1,n]\} && (1) \\ & \qquad \cup \{q_{i,j} \xrightarrow{!\alpha^{W \to L}} q_{i,j+1} \mid \\ & \qquad\quad \alpha = w_{i,j+1}, i \in [1,n], j \in [0, |w_i| - 2]\} && (2) \\ & \qquad \cup \{q_{i,|w_i|-1)} \xrightarrow{!\alpha^{W \to L}} q_f \mid \alpha = w_{i,|w_i|}, i \in [1,n]\} && (3) \\ & \qquad \cup \{q_f \xrightarrow{!i^{W \to I}} q_{i,0} \mid i \in [1,n]\} && (4) \\ & \qquad \cup \{q_f \xrightarrow{!\$^{W \to I}} q_\$\} && (5) \\ & \qquad \cup \{q_\$ \xrightarrow{!end^{W \to L}} q_e\}; && (6) \end{aligned}$$

- $\mathcal{A}_{W'} = (S_{W'}, s_0^{W'}, M, \longrightarrow_{W'})$ where:

$$\begin{aligned} & S_{W'} = \{q_0\} \cup \{q_{i,j} \mid i \in [1,n], j \in [0, |w'_i| - 1]\} \cup \{q_f, q_\$, q_e\} \\ & s_0^{W'} = q_0 \\ & \longrightarrow_{W'} = \{q_0 \xrightarrow{!i^{W' \to I}} q_{i,0} \mid i \in [1,n]\} && (7) \\ & \qquad \cup \{q_{i,j} \xrightarrow{!\alpha^{W' \to L}} q_{i,j+1} \mid \\ & \qquad\quad \alpha = w'_{i,j+1}, i \in [1,n], j \in [0, |w'_i| - 2]\} && (8) \\ & \qquad \cup \{q_{i,|w'_i|-1} \xrightarrow{!\alpha^{W' \to L}} q_f \mid \alpha = w'_{i,|w'_i|}, i \in [1,n]\} && (9) \\ & \qquad \cup \{q_f \xrightarrow{!i^{W' \to I}} q_{i,0} \mid i \in [1,n]\} && (10) \\ & \qquad \cup \{q_f \xrightarrow{!\$^{W' \to I}} q_\$\} && (11) \\ & \qquad \cup \{q_\$ \xrightarrow{!end^{W' \to L}} q_e\}; && (12) \end{aligned}$$

- $\mathcal{A}_L = (S_L, s_0^L, M, \longrightarrow_L)$ where:

$$S_L = \{q_0, q_e, q_{e'}, q_{ok}, q_*\} \cup \{q_\alpha | \alpha \in \Sigma\}$$
$$s_0^L = q_0$$
$$\begin{aligned}
\longrightarrow_L = & \{q_0 \xrightarrow{?\alpha^{W\to L}} q_\alpha \mid \alpha \in \Sigma\} && (13)\\
& \cup \{q_\alpha \xrightarrow{?\alpha'^{W'\to L}} q_0 \mid \alpha \in \Sigma\} && (14)\\
& \cup \{q_\alpha \xrightarrow{?\beta'^{W'\to L}} q_* \mid \beta \in \Sigma, \beta \neq \alpha\} \\
& \quad \cup \{q_\alpha \xrightarrow{?end'^{W'\to L}} q_*\} && (15)\\
& \cup \{q_0 \xrightarrow{?\alpha'^{W'\to L}} q_*\} \cup \{q_0 \xrightarrow{?end'^{W'\to L}} q_*\} \\
& \quad \cup \{q_\alpha \xrightarrow{?\beta^{W\to L}} q_* \mid \beta \in \Sigma\} \cup \{q_\alpha \xrightarrow{?end^{W\to L}} q_* \} \\
& \quad \cup \{q_e \xrightarrow{?\alpha'^{W'\to L}} q_* \mid \alpha \in \Sigma\} && (16)\\
& \cup \{q_* \xrightarrow{?m^{p\to L}} q_* \mid m^{p\to L} \in M\} && (17)\\
& \cup \{q_0 \xrightarrow{?end^{W\to L}} q_e \} && (18)\\
& \cup \{q_e \xrightarrow{?end'^{W'\to L}} q_{e'} \} && (19)\\
& \cup \{q_{e'} \xrightarrow{!ok^{L\to I}} q_{ok}\}; && (20)
\end{aligned}$$

- $\mathcal{A}_I = (S_I, s_0^I, M, \longrightarrow_I)$ where:

$$S_I = \{q_0, q_\$, q_{\$'}, q_*\} \cup \{q_i \mid i \in [1, n]\}$$
$$s_0^I = q_0$$
$$\begin{aligned}
\longrightarrow_I = & \{q_0 \xrightarrow{?i^{W\to I}} q_i \mid i \in [1, n]\} && (21)\\
& \cup \{q_i \xrightarrow{?i'^{W'\to I}} q_0 \mid i \in [1, n]\} && (22)\\
& \cup \{q_i \xrightarrow{?k'^{W'\to I}} q_* \mid i, k \in [1, n], i \neq k\} \\
& \quad \cup \{q_i \xrightarrow{?\$'^{W'\to I}} q_* \mid i \in [1, n]\} && (23)\\
& \cup \{q_0 \xrightarrow{?i'^{W'\to I}} q_* | \ i \in [1, n]\} \cup \{q_0 \xrightarrow{?\$'^{W'\to I}} q_*\} \\
& \quad \cup \{q_\$ \xrightarrow{?i'^{W'\to I}} q_* \mid i \in [1, n]\} \\
& \quad \cup \{q_i \xrightarrow{?k^{W\to I}} q_* | \ i, k \in [1, n]\} \\
& \quad \cup \{q_i \xrightarrow{?\$^{W\to I}} q_* \mid i \in [1, n]\} && (24)\\
& \cup \{q_* \xrightarrow{?i^{W\to I}} q_* \mid i \in [1, n]\} \cup \{q_* \xrightarrow{?i'^{W'\to I}} q_* \mid i \in [1, n]\} \\
& \quad \cup \{q_* \xrightarrow{?\$^{W\to I}} q_* \} \cup \{q_* \xrightarrow{?\$'^{W'\to I}} q_* \} && (25)
\end{aligned}$$

$$\cup \{q_* \xrightarrow{?ok^{L\to I}} q_* \} \tag{26}$$

$$\cup \{q_0 \xrightarrow{?\$^{W\to I}} q_\$ \} \tag{27}$$

$$\cup \{q_\$ \xrightarrow{?\$'^{W'\to I}} q_{\$'} \}; \tag{28}$$

- $F = \{(q_e^W, q_e^{W'}, q_{ok}^L, q_{\$'}^I)\}$.

Next, we give an informal explanation of previous definition. Please refer to Figures ?? to ?? for a better understanding.

Automata $\mathcal{A}_W$ and $\mathcal{A}_{W'}$ start in state q_0. They choose independently an index to send to $\mathcal{A}_I$ (transitions ?? and ??) and send letters of the corresponding word to $\mathcal{A}_L$ (transitions ?? and ??). For each letter of the word, each automaton sends a message and changes its state, until it reaches state q_f (transitions ?? and ??). For example, in Figure ??, $\mathcal{A}_W$ can choose the third word, sends message $3^{W\to I}$, goes in state $q_{3,0}$, sends all letters of w_3 and so it goes in state $q_{3,1}$, $q_{3,2}$, $q_{3,3}$ and finally q_f. In the state q_f, $\mathcal{A}_W$ and $\mathcal{A}_{W'}$ can, either choose a new index (transitions ?? and ??) and send the associated word, or send message \$, resp. \$', to $\mathcal{A}_I$ (transitions ?? and ??) and message *end*, resp. *end'*, to $\mathcal{A}_L$ (transitions ?? and ??), indicating the end of the sequence.

Automata $\mathcal{A}_L$ and $\mathcal{A}_I$ start in state q_0 from which they receive letters, or respectively indices, from $\mathcal{A}_W$ and $\mathcal{A}_{W'}$. For each letter, or index, from $\mathcal{A}_W$ that is in buffer b_{WL}, or b_{WI}, the automaton reads it, goes to the corresponding state (transitions ?? and ??) and reads the first letter, or index, from $\mathcal{A}_{W'}$ in buffer $b_{W'L}$, or $b_{W'I}$. Either this letter, or index, corresponds to the previous one and to the current state and the automaton goes back to state q_0 (transitions ?? and ??), or this letter, or index, does not match and the automaton goes to state q_* (transitions ?? and ??). We can see this state as a sink state. Indeed automata go in this state from any state when a message is received and does not correspond to the expected one (transitions ?? and ??). Once in q_*, both automata can receive any message (transitions ?? and ??) but they cannot leave q_*. For example, if $\mathcal{A}_W$ sends word w_3 and then $\mathcal{A}_{W'}$ sends w_3', as we can see in Figure ?? and ??, $\mathcal{A}_I$ receives $3^{W\to I}$ goes in state q_3, receives $3^{W\to I}$ and returns in state q_0. However, $\mathcal{A}_L$ receives $a^{W\to L}$, goes in state q_a, and receives $b'^{W'\to L}$ that makes it go in state q_*.

When $\mathcal{A}_L$ receive message *end* (transition ??), it waits for corresponding message *end'* from $\mathcal{A}_{W'}$. If it receives it, it goes to state $q_{e'}$ (transition ??). From this state, it can send message *ok* to $\mathcal{A}_I$ which signifies that all comparisons of letters have succeeded (transition ??).

When $\mathcal{A}_I$ receives message \$ (transition ??), it waits for corresponding message \$' from $\mathcal{A}_{W'}$. If it receives it, all comparisons of indices have succeeded (transition ??) and it goes to state $q_{\$'}$. In this state, it cannot receive message *ok* from $\mathcal{A}_L$. However, if a comparison fails, $\mathcal{A}_I$ goes to state q_* where it can receive message *ok* (transition ??).

In synchronous communication, automata $\mathcal{A}_L$ and $\mathcal{A}_I$ read messages in the same order as they are received, whatever the sender. So, the only behaviour allowing $\mathcal{A}_I$ to be in $q_{\$'}$ and $\mathcal{A}_L$ to be in $q_{e'}$ involve that $\mathcal{A}_W$ and $\mathcal{A}_{W'}$ send alternately indices to $\mathcal{A}_I$, and letters to $\mathcal{A}_L$ respectively. However, within synchronous communication, if $\mathcal{A}_I$ and $\mathcal{A}_L$ are in these state, message *ok* can not be sent because it can not be received by $\mathcal{A}_I$ in state $q_{\$'}$, and so the final global state can not be reached.

Example 4.1 (Encoding)**.** *We take PCP instance (W,W') of Example* **??**. *Encoding* $[\![\mathcal{W}, \mathcal{W}']\!]^{1-1}$ *is network* $\mathcal{N} = ((\mathcal{A}_p)_{p\in P}, M, F)$ *with* $P = (W, W', L, I)$. *Figures* **??**, **??**, **??** *and* **??** *represent these automata.*

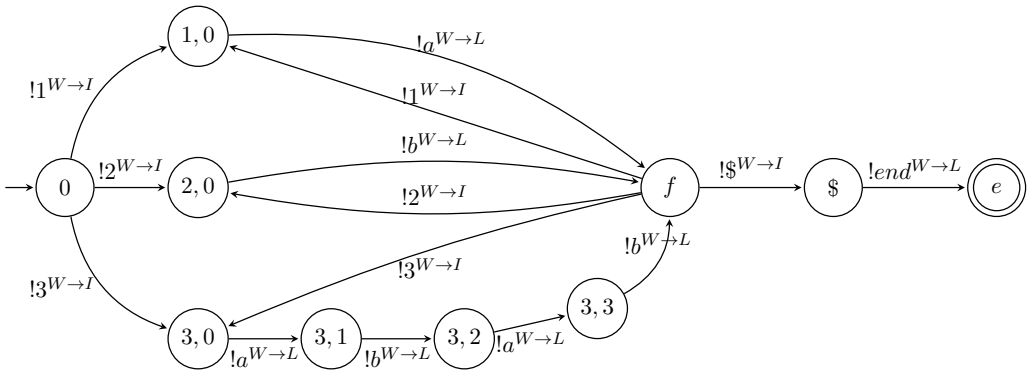


Figure 4.2: Automaton $\mathcal{A}_W$

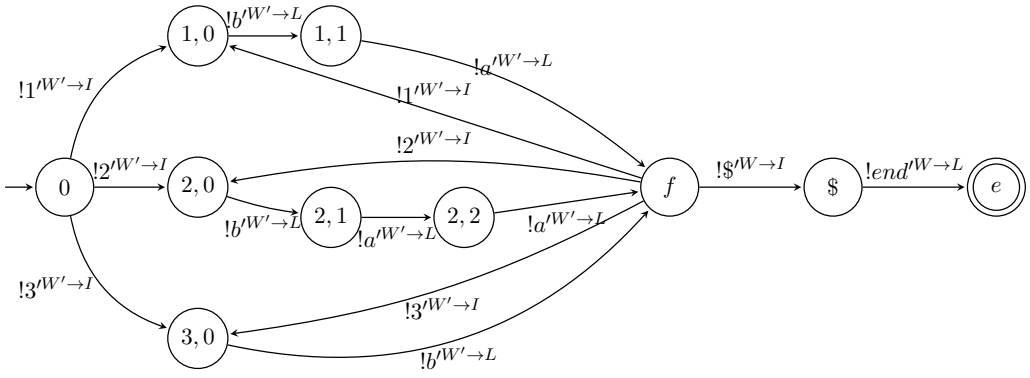


Figure 4.3: Automaton $\mathcal{A}_{W'}$

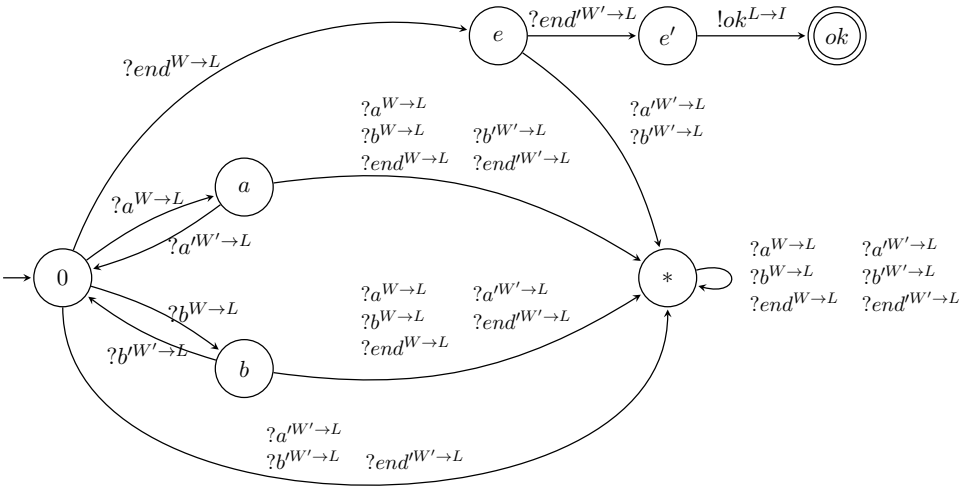


Figure 4.4: Automaton $\mathcal{A}_L$

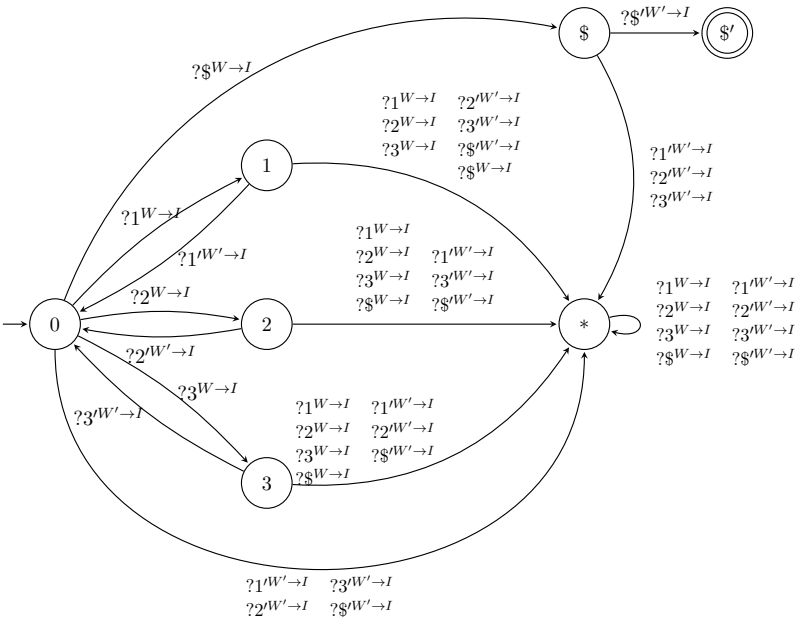


Figure 4.5: Automaton $\mathcal{A}_I$

4.2 Undecidability of Synchronizability Problem for peer-to-peer systems

We will prove that the asynchronous system, based on the encoding of a PCP instance (W,W'), is synchronizable if and only if the instance has a solution. Proofs of the results in this section can be found in Appendix **??**.

Indeed, the language associated to the synchronous system, based on the encoding of this instance, is empty. The system can not reach its global final state because of message *ok* which can not be sent since it can not be received.

Lemma 1. *Let (W, W') a PCP instance and* $[\![\mathcal{W}, \mathcal{W}']\!]^{p2p} = \mathcal{N}$ *its encoding into communicating automata, then* $\mathcal{L}(\mathcal{N}_0) = \emptyset$.

The language with a peer-to-peer communication is not empty if and only if the instance has a solution. If the language contains a final send trace, then the global final state was reached and it is possible only if comparisons of $\mathcal{A}_I$ and $\mathcal{A}_L$ succeed. In the same way, if a final send trace exists, then we can construct a solution of the instance from this.

Lemma 2. *For all PCP instance (W, W') with* $[\![\mathcal{W}, \mathcal{W}']\!]^{p2p} = \mathcal{N}$, *(W, W') has a solution if and only if* $\mathcal{L}(\mathcal{N}_{1-1^\infty}) \neq \emptyset$.

Finally, the encoding of a PCP instance is synchronizable if and only if this instance does not have a solution. Indeed, the synchronous language is always empty, and the asynchronous language contains a final send traces only if it exists a solution.

Lemma 3. *For all instance (W, W') of PCP where* $[\![\mathcal{W}, \mathcal{W}']\!]^{p2p} = \mathcal{N}$, *(W, W') has a solution if and only if* $\mathcal{L}(\mathcal{N}_0) \neq \mathcal{L}(\mathcal{N}_{1-1^\infty})$.

From these lemmas, we can establish the undecidability of the synchronizability for peer-to-peer systems.

Theorem 1. *The Synchronizability Problem is undecidable for peer-to-peer systems.*

Remark 5. *If we define all states of the network as final global state (which is equivalent to not having the notion of final global state), the Synchronizability Problem is still undecidable. Our encoding needs few modifications to establish the result.*

Indeed, only $\mathcal{A}_I$ *needs to be modified. The idea is to check the state of* $\mathcal{A}_L$ *(what we are already doing with message ok) but also the state of I. We want* $\mathcal{A}_L$ *is in state* $q_{end'}$ *and* $\mathcal{A}_I$ *is in state* $q_{\$'}$, *which means comparisons of letters and indices succeeded. To verify that,* $\mathcal{A}_L$ *send message ok from state* $q_{end'}$ *to* $\mathcal{A}_I$. $\mathcal{A}_I$ *can receive it in* q_* *but also in* $q_{\$'}$ *(see Figure* **??**, *which corresponds to Figure* **??** *where only added transitions are wrote). If it is*

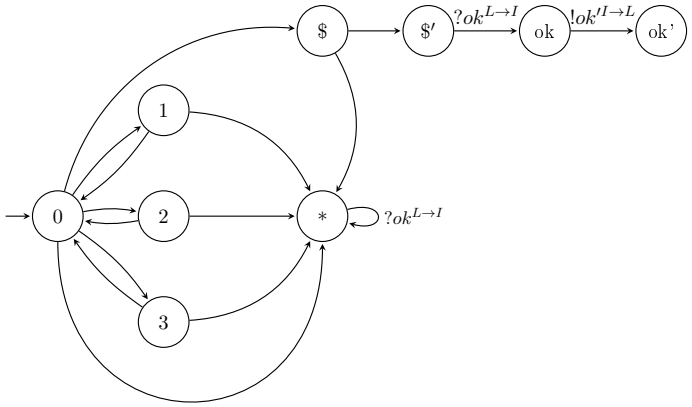


Figure 4.6: Automaton $\mathcal{A}_I$ without final state

in $q^I_{\$'}$ and it receives ok, it can send ok′ to $\mathcal{A}_L$ which is not able to read it. Indeed, there is no transitions with this label. Thus, this message can be sent only within asynchronous communication when comparisons of indices and letters succeeded, so when there exists a solution to the PCP instance.

5 Synchronizability Problem for Mailbox Systems

Counter-examples of Section **??** put us on the path of the undecidability of the synchronizability for mailbox systems. Thus, in this section, we show the undecidability of synchronizability for systems with this type of communication. As in the previous section, the proof is a reduction from PCP. We denote $\mathcal{N} = [\![\mathcal{W}, \mathcal{W}']\!]^{mail}$ the encoding of an instance (W, W'). We prove that a solution to this instance exists if and only if

$$\mathcal{L}(\mathcal{N}_0) \neq \mathcal{L}(\mathcal{N}_{*-1^\infty})$$

5.1 Mailbox Encoding of Post Correspondence Problem

The encoding in mailbox system is different from the one with peer-to-peer communication. Indeed, the fact that a same buffer can receive messages from different senders poses a number of constraints. $\mathcal{A}_W$ and $\mathcal{A}_{W'}$ have to coordinate and alternate their sendings so that $\mathcal{A}_I$ and $\mathcal{A}_L$ do not receive two consecutive messages from the same automaton.

As a counter-example, we consider instance (W,W') of PCP of Example **??** to show why the previous encoding does not allow to reduce the PCP to the Synchronizability Problem for mailbox systems. As said before, $\mathcal{A}_W$ and $\mathcal{A}_{W'}$ have to be regulated. We will write send trace t^s respecting this

constraint and allowing to reach the final global state. We know that a solution of this instance is $Sol = (2, 1, 3)$.

$$t^s = !2^{W \to I} \cdot !2'^{W' \to I} \cdot !b^{W \to L} \cdot !b'^{W' \to L} \cdot$$

($\mathcal{A}_W$ is in state q_f so to send a new letter to $\mathcal{A}_L$, it has to send a new index to $\mathcal{A}_I$.)

$$!1^{W \to I} \cdot !a^{W \to L} \cdot !a'^{W' \to L} \cdot$$

(Again, $\mathcal{A}_W$ is in state q_f so to send a new letter to $\mathcal{A}_L$, it has to send a new index to $\mathcal{A}_I$.)

$$!3^{W \to I} \cdot !a^{W \to L} \cdot !a'^{W' \to L}$$

At this state, buffer b_L of $\mathcal{A}_L$ contains: $b_L = bb'aa'aa'$; but buffer b_I of $\mathcal{A}_I$ contains: $b_I = 22'13$. So, comparisons of $\mathcal{A}_I$ will fail. Indeed, the difference of length of words makes that two indices of $\mathcal{A}_W$ have been sent consecutively.

This system can not reach its final global state, so $\mathcal{L}(\mathcal{N}_{*-1^\infty}) = \emptyset$, however a solution exists to this instance. Thus, this encoding does not allow to reduce the PCP to the Synchronizability Problem for mailbox systems.

We have to define a new encoding to mailbox system. The encoding of an instance (W,W') of PCP is a parallel composition of four automata : $\mathcal{B}_I$, $\mathcal{B}_W$, $\mathcal{B}_{W'}$, and $\mathcal{B}_L$. In this encoding, $\mathcal{B}_I$ sends the same indices to $\mathcal{B}_W$ and $\mathcal{B}_{W'}$ which sends the respective words to $\mathcal{B}_L$. $\mathcal{B}_L$ compare letters and, at the end of the run, its state allows to say if a solution exists. The topology and the buffer layout of the system is depicted in Figure **??**.

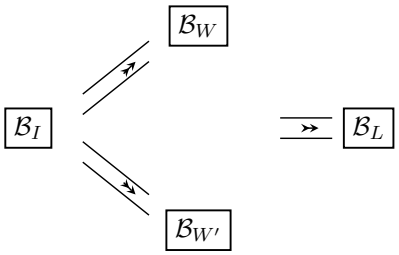


Figure 5.1: Topology of $\mathcal{N}$

More formally, we define the encoding as follows.

Definition 15 (*Encoding of PCP in mailbox system*)**.** Let (W,W') be a PCP instance over Σ, where $W = w_1, \ldots, w_n$ and $W' = w'_1, \ldots, w'_n$. The encoding of (W,W') is $[\![\mathcal{W}, \mathcal{W}']\!]^{mail} = ((\mathcal{A}_p)_{p \in P}, M, F)$ where:

- $P = \{I, W, W', L\}$;
- The set of messages is

$$M = \{i^{I\to W} \mid i \in [1,n]\} \cup \{i'^{I\to W'} \mid i \in [1,n]\} \cup \{\$^{I\to W}\} \cup \{\$'^{I\to W'}\} \cup$$
$$\{\alpha^{W\to L} \mid \alpha \in \Sigma\} \cup \{\alpha'^{W'\to L} \mid \alpha \in \Sigma\} \cup$$
$$\{end^{W\to L}\} \cup \{end'^{W'\to L}\} \cup \{e^{L\to I}\};$$

- $\mathcal{B}_I = (S_I, s_0^I, M, \longrightarrow_I)$ where:

$$S_I = \{q_0, q_\$, q_{\$'}\} \cup \{q_i \mid i \in [1,n]\}$$
$$s_0^I = q_0$$
$$\longrightarrow_I = \{q_0 \xrightarrow{!i^{I\to W}} q_i \mid i \in [1,n]\} \tag{29}$$
$$\cup \{q_i \xrightarrow{!i'^{I\to W'}} q_0 \mid i \in [1,n]\} \tag{30}$$
$$\cup \{q_0 \xrightarrow{!\$^{I\to W}} q_\$\} \tag{31}$$
$$\cup \{q_\$ \xrightarrow{!\$'^{I\to W'}} q_{\$'}\}; \tag{32}$$

- $\mathcal{B}_W = (S_W, s_0^W, M, \longrightarrow_W)$ where:

$$S_W = \{q_0, q_f, q_\$, q_e\} \cup \{q_{i,j} \mid i \in [1,n], j \in [0, |w_i| - 1]\}$$
$$s_0^W = q_0$$
$$\longrightarrow_W = \{q_0 \xrightarrow{?i^{I\to W}} q_{i,0} \mid i \in [1,n]\} \tag{33}$$
$$\cup \{q_{i,j} \xrightarrow{!\alpha^{W\to L}} q_{i,j+1} \mid \alpha = w_{i,j+1}, i \in [1,n], j \in [1, |w_i| - 2]\} \tag{34}$$
$$\cup \{q_{i,|w_i|-1} \xrightarrow{!\alpha^{W\to L}} q_f \mid \alpha = w_{i,|w_i|}, i \in [1,n]\} \tag{35}$$
$$\cup \{q_f \xrightarrow{?i^{I\to W}} q_{i,0} \mid i \in [1,n]\} \tag{36}$$
$$\cup \{q_f \xrightarrow{?\$^{I\to W}} q_\$\} \tag{37}$$
$$\cup \{q_\$ \xrightarrow{!end^{W\to L}} q_e\}; \tag{38}$$

- $\mathcal{B}_{W'} = (S_{W'}, s_0^{W'}, M, \longrightarrow_{W'})$ where:

$$S_{W'} = \{q_0, q_f, q_\$, q_e\} \cup \{q_{i,j} \mid i \in [1,n], j \in [0, |w_i| - 1]\}$$
$$s_0^{W'} = q_0$$
$$\longrightarrow_{W'} = \{q_0 \xrightarrow{?i'^{I\to W'}} q_{i,0} \mid i \in [1,n]\} \tag{39}$$
$$\cup \{q_{i,j} \xrightarrow{!\alpha'^{W'\to L}} q_{i,j+1} \mid \alpha = w'_{i,j+1}, i \in [1,n], j \in [1, |w'_i| - 2]\} \tag{40}$$

$$\cup \{q_{i,|w'_i|-1} \xrightarrow{!\alpha'^{W'\to L}} q_f \mid \alpha = w'_{i,|w'_i|}, i \in [1,n]\} \quad (41)$$

$$\cup \{q_f \xrightarrow{?i'^{I\to W'}} q_{i,0} \mid i \in [1,n]\} \quad (42)$$

$$\cup \{q_f \xrightarrow{?\$'^{I\to W'}} q_\$\} \quad (43)$$

$$\cup \{q_\$ \xrightarrow{!end'^{W'\to L}} q_e\}; \quad (44)$$

- $\mathcal{B}_L = (S_L, s_0^L, M, \longrightarrow_L)$ where:

$S_L = \{q_0, q_e, q_{e'}, q_{ok}, q_*\} \cup \{q_\alpha | \alpha \in \Sigma\}$
$s_0^L = q_0$
$\to_L = \{q_0 \xrightarrow{?\alpha^{W\to L}} q_\alpha \mid \alpha \in \Sigma\}$ (45)

$$\cup \{q_\alpha \xrightarrow{?\alpha'^{W'\to L}} q_0 \mid \alpha \in \Sigma\} \quad (46)$$

$$\cup \{q_\alpha \xrightarrow{?\beta'^{W'\to L}} q_* \mid \beta \in \Sigma, \beta \neq \alpha\}$$
$$\cup \{q_\alpha \xrightarrow{?end'^{W'\to L}} q_*\} \quad (47)$$

$$\cup \{q_0 \xrightarrow{?\alpha'^{W'\to L}} q_*\} \cup \{q_0 \xrightarrow{?end'^{W'\to L}} q_*\}$$
$$\cup \{q_\alpha \xrightarrow{?\beta^{W\to L}} q_* \mid \beta \in \Sigma\} \cup \{q_\alpha \xrightarrow{?end^{W\to L}} q_*\}$$
$$\cup \{q_e \xrightarrow{?\alpha^{W\to L}} q_* \mid \alpha \in \Sigma\}$$
$$\cup \{q_e \xrightarrow{?\alpha'^{W'\to L}} q_* \mid \alpha \in \Sigma\} \quad (48)$$

$$\cup \{q_* \xrightarrow{?\alpha^{W\to L}} q_* \mid \alpha \in \Sigma\} \cup \{q_* \xrightarrow{?\alpha'^{W'\to L}} q_* \mid \alpha \in \Sigma\}$$
$$\cup \{q_* \xrightarrow{?end^{W\to L}} q_*\} \cup \{q_* \xrightarrow{?end'^{W'\to L}} q_*\} \quad (49)$$

$$\cup \{q_0 \xrightarrow{?end^{W\to L}} q_e\} \quad (50)$$

$$\cup \{q_e \xrightarrow{?end'^{W'\to I}} q_{e'}\} \quad (51)$$

$$\cup \{q_{e'} \xrightarrow{!ok^{L\to I}} q_{ok}\}; \quad (52)$$

- $F = \{(q_{\$'}^I, q_e^W, q_e^{W'}, q_{ok}^L)\}$.

Next, we give an informal explanation of previous definition. Please refer to Figures **??** to **??** for a better understanding.

Automaton $\mathcal{B}_I$ chooses an index, sends it to $\mathcal{B}_W$ (transitions **??**) and goes to the corresponding state. Then it sends the same index to $\mathcal{B}_{W'}$ and returns in its initial state (transitions **??**). To finish the sequence of indices, it send messages \$ (transition **??**) and \$′ (transition **??**). For example, in

Figure ??, it can send message $2^{I \to W}$ and goes to state q_2. Then, it sends message $2'^{I \to W'}$ and returns in state q_0. When the sequence of indices is finished, it sends $\$^{I \to W}$, goes in state $q_\$$, sends $\$'^{I \to W'}$ and goes in state $q_{\$'}$.

Automaton $\mathcal{B}_W$, and respectively $\mathcal{B}_{W'}$, receives an index from $\mathcal{B}_I$ (transitions ?? and ??) and send the letters of the corresponding word to $\mathcal{B}_L$ (transitions ?? and ??). Note that the send trace allowing to reach the global final state alternate a letter of $\mathcal{B}_W$ with a letter of $\mathcal{B}_{W'}$. At the end of each word, the automaton is in state q_f (transitions ?? and ??) in which it can either receive a new index (transitions ?? and ??), or receive message \$, respectively \$' (transitions ?? and ??). If it receives \$, it sends message *end*, respectively *end'*, to $\mathcal{B}_L$ (transitions ?? and ??). For example, if $\mathcal{B}_I$ begins by sending index 2, as we can see in Figure ?? and ??, $\mathcal{B}_W$ and $\mathcal{B}_{W'}$ go in state $q_{2,0}$. $\mathcal{B}_W$ sends $b^{W \to L}$ and goes to q_f, then $\mathcal{B}_{W'}$ sends $b'^{W' \to L}$ and goes to state $q_{2,1}$. It has to wait the next letter of $\mathcal{B}_W$ before sending $a'^{W' \to L}$ and going in state $q_{2,2}$, or $\mathcal{B}_L$ would receive two letters from $\mathcal{B}_{W'}$ consecutively and go in state q_*.

Indeed, automaton $\mathcal{B}_L$ receives a letter from $\mathcal{B}_W$ and goes to the corresponding state (transitions ??). If it receives the same letter from $\mathcal{B}_{W'}$, then it goes back to state q_0 (transitions ??), otherwise it goes to state q_* (transitions ??) which, as in previous encoding, can be seen as a sink state (transitions ?? and ??). For example, as we can see in Figure ??, Figure ?? and Figure ??, if $\mathcal{B}_W$ and $\mathcal{B}_{W'}$ received index 2, $\mathcal{B}_W$ sends $b^{W \to L}$, $\mathcal{B}_L$ receives it and goes in state q_b. Then, $\mathcal{B}_{W'}$ sends $b'^{W' \to L}$ and $\mathcal{B}_L$ returns in q_0. As we said previously, if $\mathcal{B}_{W'}$ sends $a'^{W' \to L}$ immediately, $\mathcal{B}_L$ goes in state q_*.

If all comparisons of letters of $\mathcal{B}_W$ and $\mathcal{B}_{W'}$ succeed until messages *end* (transition ??) and *end'* (transition ??), $\mathcal{B}_L$ is in state $q_{e'}$. If the system is asynchronous, then $\mathcal{B}_L$ can send message *ok*, otherwise if the system is synchronous, then $\mathcal{B}_L$ can send message *ok* only if $\mathcal{B}_I$ is in state q_* from which it can receive *ok* (transition ??).

Example 5.1 (Encoding)**.** *We take instance (W, W') of PCP of Example* ??*. Encoding* $[\![\mathcal{W}, \mathcal{W}']\!]^{1-1}$ *is network* $\mathcal{N} = ((\mathcal{B}_p)_{p \in P}, M, F)$ *with* $P = (I, W, W', L)$*. Figures* ??*,* ??*,* ?? *and* ?? *represent these automata.*

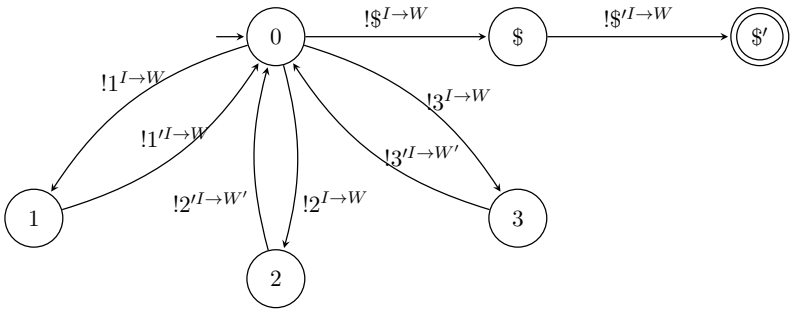


Figure 5.2: Automaton $\mathcal{B}_I$

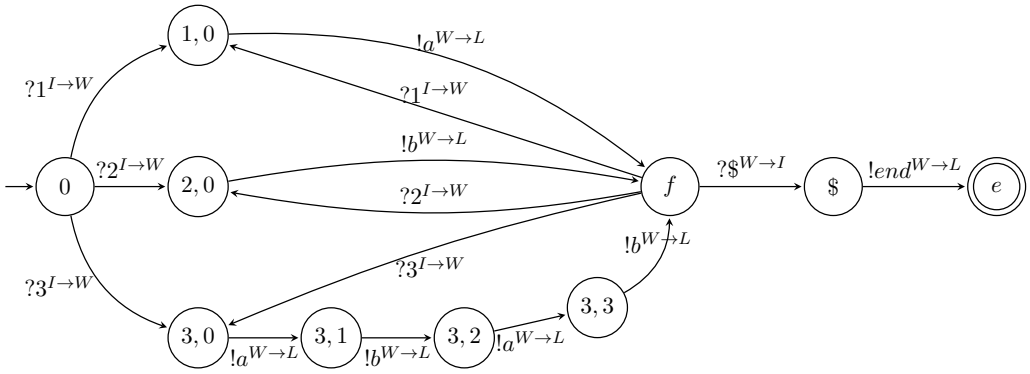


Figure 5.3: Automaton $\mathcal{B}_W$

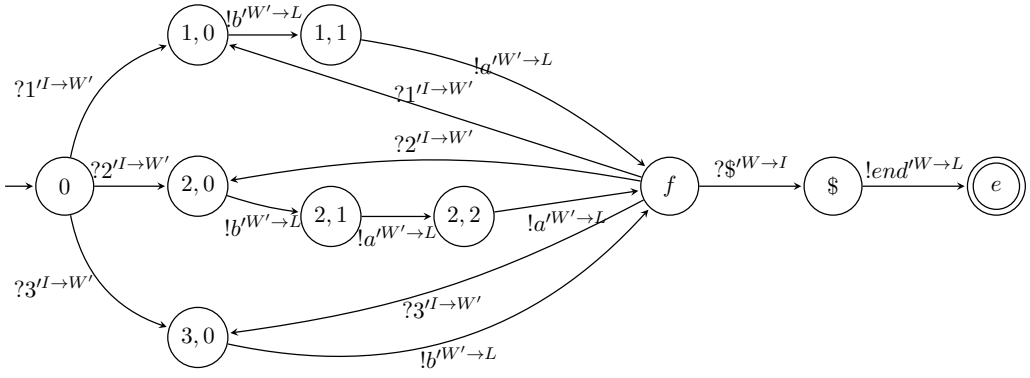


Figure 5.4: Automaton $\mathcal{B}_{W'}$

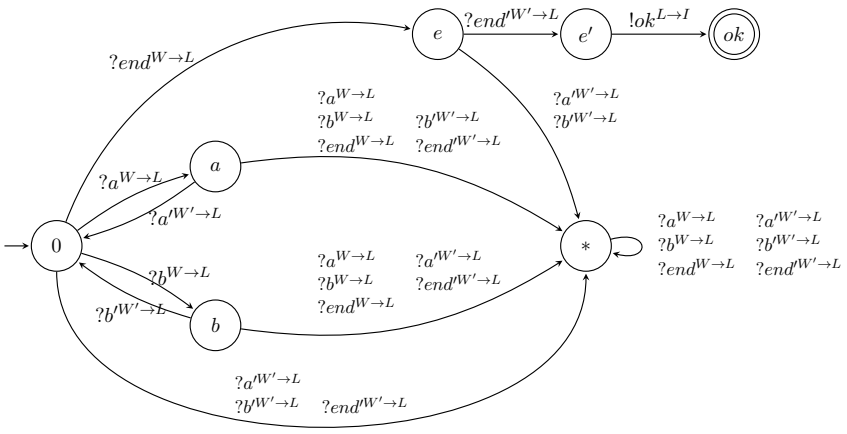


Figure 5.5: Automaton $\mathcal{B}_L$

5.2 Undecidability of Synchronizability Problem for mailbox systems

We will prove that the asynchronous system, based on the encoding of a PCP instance (W,W'), is synchronizable if and only if the instance has a solution. Proofs of the results in this section can be found in Appendix **??**.

Indeed, the language associated to the synchronous system, based on the encoding of this instance, is empty. The system can not reach its global final state because of message *ok* which can not be sent since it can not be received.

Lemma 4. *Let (W,W') an instance of PCP and* $[\![\mathcal{W}, \mathcal{W}']\!]^{mail} = \mathcal{N}$ *its encoding into communicating automata, then* $\mathcal{L}(\mathcal{N}_0) = \emptyset$.

Within asynchronous communication, this message *ok* can be send only if the encoded instance of PCP has a solution. So the global final state can be reached, and the language is not empty, only if it exists a solution.

Lemma 5. *For all instance (W,W') of PCP where* $[\![\mathcal{W}, \mathcal{W}']\!]^{mail} = \mathcal{N}$, *(W,W') has a solution if and only if* $\mathcal{L}(\mathcal{N}_{*-1^\infty}) \neq \emptyset$.

Therefore, the system is synchronizable if and only if the encoded instance does not have solution.

Lemma 6. *For all instance (W,W') of PCP where* $[\![\mathcal{W}, \mathcal{W}']\!]^{mail} = \mathcal{N}$, *(W,W') has a solution if and only if* $\mathcal{L}(\mathcal{N}_0) \neq \mathcal{L}(\mathcal{N}_{*-1^\infty})$.

Thus, from these lemmas, we can establish the undecidability of synchronizability for mailbox systems.

Theorem 2. *The Synchronizability Problem is undecidable for mailbox systems.*

Remark 6. *In mailbox communication, we can not work without the notion of global final state, as in peer-to-peer communication. Synchronizability without global final states for mailbox systems remains an open problem.*

6 Related Works

Several authors have looked at the notion of synchronizability or other notions related to it.

In [?], Basu and Bultan defined synchronizability over language based on send traces only, which amounts with our definitions to all states being final. In this paper, the authors establish a hierarchy of synchronizability and they claim that, for all networks $\mathcal{N}$:

$$\mathcal{N}_{type^\infty} \textit{ is synchronizable if and only if} \mathcal{L}(\mathcal{N}_0) = \mathcal{L}(\mathcal{N}_{type^\infty})$$

for $type \in \{1-1; *-1\}$. They also study other types of asynchrony, shared memory systems and bag systems where there is a unique FIFO buffer and respectively a unique buffer which is not a FIFO, to store all messages.

However, in [?], Finkel and Lozes prove this first assertion is false, with counter-examples for systems with both peer-to-peer (1 - 1) and mailbox (* - 1) communication (see Figure ?? for mailbox systems). Furthermore, they prove that the Synchronizability Problem is undecidable for peer-to-peer systems.

Other works, even if they differ by the type of trace that is observed and by the modelling of systems, are close to ours. Genest et al. in [?] studied deadlock-free distributed systems, with a peer-to-peer communication. Here, they are not interested in send traces but in Mazurkiewicz traces (see [?] for more details). Intuitively, these traces keep both sendings and receptions but allow permutations between some actions. In order to model communications, they use Message Sequences Charts (MSC). They also use the notions of final state to study systems. More precisely, they seek to know if these systems are universally or existentially bounded, and specify when this is decidable. A system is universally bounded if it exists a bound B such that, regardless of the schedule of the actions, every run can be executed with a maximum of B transitory messages in the system, and existentially bounded if every runs can be rescheduled (according to the definition of Mazurkiewicz trace) such that they can be executed with a maximum of B transitory messages in the system. This last property is close to the notion of stability, indeed if a system is existentially bounded, it means that all traces can be executed respecting a bound for the buffers.

In [?], Bouajjani et al. studied Mazurkiewicz traces on mailbox systems and model them with MSC. The observed property is the k-synchronizability,

which is close to the notion of stability. They establish properties on conflict graphs characterizing k-synchronizable traces and also study the decidability of the k-synchronizability.

7 Conclusion

During my internship, we focussed on the property of synchronizability. In a synchronizable system, the behaviour is independent from the type of communication. This entails that a synchronizable system can be analysed with tools for synchronous systems that are, in general, more powerful and efficient (e.g., properties for synchronous systems are in general decidable).

More precisely, we looked at the decidability of the synchronizability problem, first for peer-to-peer systems, and then for mailbox systems. For the former, we proved the undecidability of synchronizability with an alternative proof than the one presented in [**?**]. For the latter, we showed the undecidability of synchronizability with the additional constraint that not all states are final.

Deciding synchronizability for mailbox systems in the general case remains an open problem, that will be the focus of my first phd year. More broadly, other questions would need to be addressed. It would be interesting to know if classes of systems for which the problem is decidable exist, and so to find conditions that would imply the synchronizability of the system by construction. Finally, we might also ask what properties could be induced by the synchronizability of a system, like for instance deadlock freedom.

Appendix A Proofs of Section ??

Lemma 1. *Let (W,W') a PCP instance and* $[\![\mathcal{W},\mathcal{W}']\!]^{p2p} = \mathcal{N}$ *its encoding into communicating automata, then* $\mathcal{L}(\mathcal{N}_0) = \emptyset$.

Proof. By contradiction, we suppose $\mathcal{L}(\mathcal{N}_0) \neq \emptyset$. Thus let t^f a final send trace such that $t^f \in \mathcal{L}(\mathcal{N}_0)$. As $(q_e^W, q_e^{W'}, q_{ok}^L, q_{\$'}^I)$ is the unique final global state and by definition of final send trace, we have: $t^f = act_1 \cdot act_2 \cdot \ldots \cdot act_e$ such that

$$((q_0^W, q_0^{W'}, q_0^L, q_0^I), B^\emptyset) \xrightarrow[0]{act_1} C_1 \xrightarrow[0]{act_2} \ldots \xrightarrow[0]{act_e} ((q_e^W, q_e^{W'}, q_{ok}^L, q_{\$'}^I), B^\emptyset)$$

By construction, there is a unique transition in $\mathcal{A}_L$ to reach state q_{ok}^L: $q_{e'}^L \xrightarrow{!ok^{L\to I}}_L q_{ok}^L$ so $\exists act_k \in t^f$ such that $act_k = !ok^{L\to I}$. The communication being synchronous, a sending must occur together with its reception (see rule (0 SEND)). By construction, there is a unique transition in $\mathcal{A}_I$ for the reception of message *ok*: $q_*^I \xrightarrow{?ok^{L\to I}}_I q_*^I$. However, there is no transition from q_*^I to $q_{\$'}^I$. So, after the sending of message *ok*, the global final state is not reachable. Thus, t^f can not exist and $\mathcal{L}(\mathcal{N}_0) = \emptyset$. □

Lemma 2. *For all PCP instance (W,W') with* $[\![\mathcal{W},\mathcal{W}']\!]^{p2p} = \mathcal{N}$, *(W,W') has a solution if and only if* $\mathcal{L}(\mathcal{N}_{1-1^\infty}) \neq \emptyset$.

Proof. $\Rightarrow$

Let $\mathrm{Sol}_{(\mathrm{W,W'})} = (i_1, i_2, \ldots, i_m)$ be a solution of (W,W'). For each index $k \in \mathrm{Sol}_{(\mathrm{W,W'})}$, corresponding words in W and W' are
$w_k = w_{k,1}w_{k,2}\ldots w_{k,|w_k|}$ and $w'_k = w'_{k,1}w'_{k,2}\ldots w'_{k,|w'_k|}$ respectively.

Take a send trace t^s of the following form:

$$\begin{aligned}
t^s = {} & !i_1^{W\to I} \cdot !w_{i_1,1}^{W\to L} \cdot !w_{i_1,2}^{W\to L} \cdot \ldots \cdot !w_{i_1,|w_{i_1}|}^{W\to L} \cdot \\
& !i_2^{W\to I} \cdot !w_{i_2,1}^{W\to L} \cdot !!w_{i_2,2}^{W\to L} \cdot \ldots \cdot !w_{i_2,|w_{i_2}|}^{W\to L} \cdot \\
& \ldots \\
& !i_m^{W\to I} \cdot !w_{i_m,1}^{W\to L} \cdot !w_{i_m,2}^{W\to L} \cdot \ldots \cdot !w_{i_m,|w_{i_m}|}^{W\to L} \cdot !\$^{W\to I} \cdot !end^{W\to L} \cdot \\
& !i_1'^{W'\to I} \cdot !w_{i_1,1}'^{W'\to L} \cdot !w_{i_1,2}'^{W'\to L} \cdot \ldots \cdot !w_{i_1,|w'_{i_1}|}'^{W'\to L} \cdot \\
& !i_2'^{W'\to I} \cdot !w_{i_2,1}'^{W'\to L} \cdot !w_{i_2,2}'^{W'\to L} \cdot \ldots \cdot !w_{i_2,|w'_{i_2}|}'^{W'\to L} \cdot \\
& \ldots \\
& !i_m'^{W'\to I} \cdot !w_{i_m,1}'^{W'\to L} \cdot !w_{i_m,2}'^{W'\to L} \cdot \ldots \cdot !w_{i_m,|w'_{i_m}|}'^{W'\to L} \cdot !\$'^{W'\to I} \cdot !end'^{W'\to L} \cdot \\
& !ok^{L\to I}
\end{aligned}$$

We show that $t^s \in \mathcal{L}(\mathcal{N}_{1-1^\infty})$. Trace t^s corresponds to the following run. First, $\mathcal{A}_W$ sends indices included in $[i_1, i_m]$ and corresponding words in W.

$$((q_0^W, q_0^{W'}, q_0^L, q_0^I), B^{\emptyset}) \xrightarrow[1-1^{\infty}]{!i_1^{W\to I}} ((q_{i_1,0}^W, q_0^{W'}, q_0^L, q_0^I), B_1 = B^{\emptyset}\{b_{WI}/b_{WI}\cdot i_1\})$$

$$\xrightarrow[1-1^{\infty}]{!w_{i_1,1}^{W\to L}} ((q_{i_1,1}^W, q_0^{W'}, q_0^L, q_0^I), B_{1'} = B_1\{b_{WL}/b_{WL}\cdot w_{i_1,1}\})$$
$$\xrightarrow[1-1^{\infty}]{!w_{i_1,2}^{W\to L}} ((q_{i_1,2}^W, q_0^{W'}, q_0^L, q_0^I), B_{1'}\{b_{WL}/b_{WL}\cdot w_{i_1,2}\})$$
$$\dots$$

$$((q_{i_1,|w_{i_1}|-2}^W, q_0^{W'}, q_0^L, q_0^I), B_2)$$

$$\xrightarrow[1-1^{\infty}]{!w_{i_1,|w_{i_1}|-1}^{W\to L}} ((q_{i_1,|w_{i_1}|-1}^W, q_0^{W'}, q_0^L, q_0^I), B_{2'} = B_2\{b_{WL}/b_{WL}\cdot w_{i_1,|w_{i_1}|-1}\})$$
$$\xrightarrow[1-1^{\infty}]{!w_{i_1,|w_{i_1}|}^{W\to L}} ((q_f^W, q_0^{W'}, q_0^L, q_0^I), B_{2'}\{b_{WL}/b_{WL}\cdot w_{i_1,|w_{i_1}|}\})$$
$$\dots$$

$$((q_f^W, q_0^{W'}, q_0^L, q_0^I), B_3) \xrightarrow[1-1^{\infty}]{!i_m^{W\to I}} ((q_{i_m,0}^W, q_0^{W'}, q_0^L, q_0^I), B_{3'} = B_3\{b_{WI}/b_{WI}\cdot i_m, 1\})$$

$$\xrightarrow[1-1^{\infty}]{!w_{i_m,1}^{W\to L}} ((q_{i_m,1}^W, q_0^{W'}, q_0^L, q_0^I), B_{3''} = B_{3'}\{b_{WL}/b_{WL}\cdot w_{i_m,1}\})$$
$$\xrightarrow[1-1^{\infty}]{!w_{i_m,2}^{W\to L}} ((q_{i_m,2}^W, q_0^{W'}, q_0^L, q_0^I), B_{3''}\{b_{WL}/b_{WL}\cdot w_{i_m,2}\})$$
$$\dots$$

$$((q_{i_m,|w_{i_m}|-2}^W, q_0^{W'}, q_0^L, q_0^I), B_4) \xrightarrow[1-1^{\infty}]{!w_{i_m,|w_{i_m}|-1}^{W\to L}}$$
$$((q_{i_m,|w_{i_m}|-1}^W, q_0^{W'}, q_0^L, q_0^I), B_{4'} = B_4\{b_{WL}/b_{WL}\cdot w_{i_m,|w_{i_m}|-1}\})$$

$$\xrightarrow[1-1^{\infty}]{!w_{i_m,|w_{i_m}|}^{W\to L}} ((q_f^W, q_0^{W'}, q_0^L, q_0^I), B_{4''} = B_{4'}\{b_{WL}/b_{WL}\cdot w_{i_m,|w_{i_m}|}\})$$

Next, it sends messages \$ and *end* indicating to $\mathcal{A}_I$ and $\mathcal{A}_L$ the end of messages of $\mathcal{A}_W$.

$$\xrightarrow[1-1^{\infty}]{!\$^{W\to I}} ((q_{\$}^W, q_0^{W'}, q_0^L, q_0^I), B_5 = B_{4''}\{b_{WI}/b_{WI}\cdot \$\})$$
$$\xrightarrow[1-1^{\infty}]{!end^{W\to L}} ((q_e^W, q_0^{W'}, q_0^L, q_0^I), B_6 = B_5\{b_{WL}/b_{WL}\cdot end\})$$

At this state, buffers b_{WI} and b_{WL} are filled with indices and letters of words in W respectively. Next it is the turn of $\mathcal{A}_{W'}$.

$$\xrightarrow[1-1^{\infty}]{!i_1'^{W'\to I}}((q_e^W, q_{i_1,0}^{W'}, q_0^L, q_0^I), B_{6'} = B_6\{b_{W'I}/b_{W'I}\cdot i_1'\})$$

$$\xrightarrow[1-1^{\infty}]{!w_{i_1,1}'^{W'\to L}}((q_e^W, q_{i_1,1}^{W'}, q_0^L, q_0^I), B_{6''} = B_{6'}\{b_{W'L}/b_{W'L}\cdot w_{i_1,1}'\})$$

$$\xrightarrow[1-1^{\infty}]{!w_{i_1,2}'^{W'\to L}}((q_e^W, q_{i_1,2}^{W'}, q_0^L, q_0^I), B_{6''}\{b_{W'L}/b_{W'L}\cdot w_{i_1,2}'\})$$

$$\dots$$

$$((q_e^W, q_{i_1,|w_{i_1}'|-2}^{W'}, q_0^L, q_0^I), B_7)$$

$$\xrightarrow[1-1^{\infty}]{!w_{i_1,|w_{i_1}'|-1}'^{W'\to L}}((q_e^W, q_{i_1,|w_{i_1}'|-1}^{W'}, q_0^L, q_0^I), B_{7'} = B_7\{b_{W'L}/b_{W'L}\cdot w_{i_1,|w_{i_1}'|-1}'\})$$

$$\xrightarrow[1-1^{\infty}]{!w_{i_1,|w_{i_1}'|}'^{W'\to L}}((q_e^W, q_f^{W'}, q_0^L, q_0^I), B_{7'}\{b_{W'L}/b_{W'L}\cdot w_{i_1,|w_{i_1}'|}'\})$$

$$\dots$$

$$((q_e^W, q_f^{W'}, q_0^L, q_0^I), B_8)\xrightarrow[1-1^{\infty}]{!i_m'^{W'\to I}}((q_e^W, q_{i_m,0}^{W'}, q_0^L, q_0^I), B_{8'} = B_8\{b_{W'I}/b_{W'I}\cdot i_m'\})$$

$$\xrightarrow[1-1^{\infty}]{!w_{i_m,1}'^{W'\to L}}((q_e^W, q_{i_m,1}^{W'}, q_0^L, q_0^I), B_{8''} = B_{8'}\{b_{W'L}/b_{W'L}\cdot w_{i_m,1}'\})$$

$$\xrightarrow[1-1^{\infty}]{!w_{i_m,2}'^{W'\to L}}((q_e^W, q_{i_m,2}^{W'}, q_0^L, q_0^I), B_{8''}\{b_{W'L}/b_{W'L}\cdot w_{i_m,2}'\})$$

$$\dots$$

$$((q_e^W, q_{i_m,|w_{i_m}'|-2}^{W'}, q_0^L, q_0^I), B_9)\xrightarrow[1-1^{\infty}]{!w_{i_m,|w_{i_m}'|-1}'^{W'\to L}}$$

$$((q_e^W, q_{i_m,|w_{i_m}'|-1}^{W'}, q_0^L, q_0^I), B_{9'} = B_9\{b_{W'L}/b_{W'L}\cdot w_{i_m,|w_{i_m}'|-1}'\})$$

$$\xrightarrow[1-1^{\infty}]{!w_{i_m,|w_{i_m}'|}'^{W'\to L}}((q_e^W, q_f^{W'}, q_0^L, q_0^I), B_{9''} = B_{9'}\{b_{W'L}/b_{W'L}\cdot w_{i_m,|w_{i_m}'|}'\})$$

$$\xrightarrow[1-1^{\infty}]{!\$'^{W'\to I}}((q_e^W, q_{\$}^{W'}, q_0^L, q_0^I), B_{10} = B_{9''}\{b_{W'I}/b_{W'I}\cdot \$'\})$$

$$\xrightarrow[1-1^{\infty}]{!end'^{W'\to L}}((q_e^W, q_e^{W'}, q_0^L, q_0^I), B_{11} = B_{10}\{b_{W'L}/b_{W'L}\cdot end'\})$$

At this state, buffers $b_{W'I}$ and $b_{W'L}$ are filled with indices and letters of words in W', respectively. Now the network proceeds by comparing indices.

$((q_e^W, q_e^{W'}, q_0^L, q_0^I), B_{11})$

$$\xrightarrow[1-1^\infty]{?i_1^{W\to I}}((q_e^W, q_e^{W'}, q_0^L, q_{i_1}^I), B_{11'} = B_{11}\{b_{WI} = i_1 \cdot b'_{WI}/b'_{WI}\})$$

$$\xrightarrow[1-1^\infty]{?i_1'^{W'\to I}}((q_e^W, q_e^{W'}, q_0^L, q_0^I), B_{11''} = B_{11'}\{b_{W'I} = i'_1 \cdot b'_{W'I}/b'_{W'I}\})$$

$$\xrightarrow[1-1^\infty]{?i_2^{W\to I}}((q_e^W, q_e^{W'}, q_0^L, q_{i_2}^I), B_{11'''} = B_{11''}\{b_{WI} = i_2 \cdot b'_{WI}/b'_{WI}\})$$

$$\xrightarrow[1-1^\infty]{?i_2'^{W'\to I}}((q_e^W, q_e^{W'}, q_0^L, q_0^I), B_{11''''} = B_{11'''}\{b_{W'I} = i'_2 \cdot b'_{W'I}/b'_{W'I}\})$$

$\dots$

$((q_e^W, q_e^{W'}, q_0^L, q_0^I), B_{12})$

$$\xrightarrow[1-1^\infty]{?i_m^{W\to I}}((q_e^W, q_e^{W'}, q_0^L, q_{i_m}^I), B_{12'} = b_{12}\{b_{WI} = i_m \cdot b'_{WI}/b'_{WI}\})$$

$$\xrightarrow[1-1^\infty]{?i_m'^{W'\to I}}((q_e^W, q_e^{W'}, q_0^L, q_0^I), B_{12''} = B_{12'}\{b_{WI} = i'_m \cdot b'_{W'I}/b'_{W'I}\})$$

$$\xrightarrow[1-1^\infty]{?\$^{W\to I}}((q_e^W, q_e^{W'}, q_0^L, q_\$^I), B_{12'''} = B_{12''}\{b_{WI} = \$/\emptyset\})$$

$$\xrightarrow[1-1^\infty]{?\$'^{W'\to I}}((q_e^W, q_e^{W'}, q_0^L, q_{\$'}^I), B_{13} = B_{12'''}\{b_{W'I} = \$'/\emptyset\})$$

At this state, buffers b_{WI} and $b_{W'I}$ are empty and comparison of indices by $\mathcal{A}_I$ has succeeded, as witnessed by state $q_{\$'}^I$. If this was not the case, the automaton would be in state q_*^I. The same task is carried on letters:

$((q_e^W, q_e^{W'}, q_0^L, q_{\$'}^I), B_{13})$

$$\xrightarrow[1-1^\infty]{?w_{i_1,1}^{W\to L}}((q_e^W, q_e^{W'}, q_{w_{i_1,1}}^L, q_{\$'}^I), B_{13'} = B_{13}\{b_{WL} = w_{i_1,1} \cdot b'_{WL}/b'_{WL}\})$$

$$\xrightarrow[1-1^\infty]{?w_{i_1,1}'^{W'\to L}}((q_e^W, q_e^{W'}, q_0^L, q_{\$'}^I), B_{14} = B_{13'}\{b_{WL} = w'_{i_1,1}.b'_{W'L}/b'_{W'L}\})$$

$\dots$

$((q_e^W, q_e^{W'}, q_0^L, q_{\$'}^I), B_{14})$

$$\xrightarrow[1-1^\infty]{?w_{i_m,|w_{i_m}|}^{W\to L}}((q_e^W, q_e^{W'}, q_{w_{i_m,|w_{i_m}|}}^L, q_{\$'}^I), B_{14'} = B_{14}\{b_{WL} = w_{i_m,|w_{i_m}|} \cdot b'_{WL}/b'_{WL}\})$$

$$\xrightarrow[1-1^\infty]{?w_{i_m,|w'_{i_m}|}'^{W'\to L}}((q_e^W, q_e^{W'}, q_0^L, q_{\$'}^I), B_{14''} = B_{14'}\{b_{WL} = w'_{i_m,|w'_{i_m}|}.b'_{W'L}/b'_{W'L}\})$$

$\xrightarrow[1-1^\infty]{?end^{W\to L}}((q_e^W, q_e^{W'}, q_e^L, q_{\$'}^I), B_{15} = B_{14}\{b_{WL} = end/\emptyset\})$

$\xrightarrow[1-1^\infty]{?end'^{W'\to L}}((q_e^W, q_e^{W'}, q_{e'}^L, q_{\$'}^I), B_{16} = B_{15}\{b_{W'L} = end'/\emptyset\})$

At this state, buffers b_{WL} and $b_{W'L}$ are empty and comparison of letters by $\mathcal{A}_L$ has succeeded, as witnessed by its state $q_{e'}^L$. If this was not the case, the automaton would be in state q_*^L. Finally:

$$((q_e^W, q_e^{W'}, q_e^L, q_{\$'}^I), B_{16}) \xrightarrow[1-1^\infty]{!ok^{L\to I}} ((q_e^W, q_e^{W'}, q_{ok}^L, q_{\$'}^I), B_{16}\{b_{LI}/b_{LI} \cdot ok\})$$

Global state $(q_e^W, q_e^{W'}, q_{ok}^L, q_{\$'}^I)$ reached with this run is the final global state of the network and $t \in \mathcal{L}(\mathcal{N}_{1-1^\infty})$. Thus $\mathcal{L}(\mathcal{N}_{1-1^\infty}) \neq \emptyset$, concluding this side of the proof.

$\Leftarrow$

If $\mathcal{L}(\mathcal{N}_{1-1^\infty}) \neq \emptyset$ then $\exists\ t \in \mathcal{L}(\mathcal{N}_{1-1^\infty})$. As $(q_e^W, q_e^{W'}, q_{ok}^L, q_{\$'}^I)$ is the unique final global state and by definition of language, t is a final send trace and we have: $t = act_1 act_2 \dots act_e$ such that

$$((q_0^W, q_0^{W'}, q_0^L, q_0^I), B^\emptyset) \xrightarrow[0]{act_1} C_1 \xrightarrow[0]{act_2} \dots \xrightarrow[0]{act_e} ((q_e^W, q_e^{W'}, q_{ok}^L, q_{\$'}^I), B_e)$$

Notice that in order to reach q_e^W and $q_e^{W'}$,

- $\exists\ act_i, act_{i'} \in t$, $i, i' \in [1, e]$, where $act_i = !j^{W\to I}$ and $act_{i'} = !j'^{W'\to I}$, $j, j' \in [1, n]$ and $j = j'$, i.e., $\mathcal{A}_W$ and $\mathcal{A}_{W'}$ have sent at least one index, with the same index (and the associated word),
- $\exists\ act_f, act_{f'}, act_g, act_{g'} \in t$, $f, f', g, g' \in [1, e]$, $i < f < g$, $i' < f' < g'$, where $act_f = !\$^{W\to I}$, $act_{f'} = !\$'^{W'\to I}$, $act_g = !end^{W\to L}$, $act_{g'} = !end'^{W'\to L}$.

Moreover, to reach $q_\$^I$, we have that:

- $\exists k \in \mathbb{N}^+$ such that t contains $m_1^{W\to I},\ m_2^{W\to I}, \dots,\ m_k^{W\to I}$ in this order and $m_{1'}^{W'\to I},\ m_{2'}^{W'\to I}, \dots,\ m_{k'}^{W'\to I}$ in this order,
- $\forall i, i' \in [1, k]$, $m_i^{W\to I} = j^{W\to I}$ and $m_{i'}^{W'\to I} = j'^{W'\to I}$, with $j \in [1, n]$, i.e., the sequences of indices sent by $\mathcal{A}_W$ and $\mathcal{A}_{W'}$ to $\mathcal{A}_I$ must be equal.

Finally, to reach q_{ok}^L, we deduce that:

- $\exists l \in \mathbb{N}^+$ such that t contains $m_1^{W\to L},\ m_2^{W\to L}, \dots,\ m_l^{W\to L}$ in this order and $m_1'^{W'\to L},\ m_2'^{W'\to L}, \dots,\ m_l'^{W'\to L}$ in this order,
- $\forall i \in [1, l]$, $m_i^{W\to I} = \alpha^{W\to I}$ and $m_i'^{W'\to L} = \alpha'^{W'\to L}$, with $\alpha \in \Sigma$, i.e., the sequences of letters sent by $\mathcal{A}_W$ and $\mathcal{A}_{W'}$ to $\mathcal{A}_L$ must be equal.

- $\exists act_z \in t$ such that $act_z = ok^{L \to I}$ and $\mathcal{A}_L$ can reach q^L_{ok}.

Then, if the network is in a final global state, indices and letters sent by $\mathcal{A}_W$ and $\mathcal{A}_{W'}$ are equal. So trace t contains a solution of instance (W,W') that can be found by extracting the messages sent by $\mathcal{A}_W$ to $\mathcal{A}_I$:

$$Sol_{(\mathrm{W,W'})} = (m_1^{W \to I}, m_2^{W \to I}, \ldots, m_k^{W \to I})$$

□

Lemma 3. *For all instance (W,W') of PCP where $[\![\mathcal{W}, \mathcal{W}']\!]^{p2p} = \mathcal{N}$, (W,W') has a solution if and only if $\mathcal{L}(\mathcal{N}_0) \neq \mathcal{L}(\mathcal{N}_{1-1^\infty})$.*

Proof. Let (W,W') be an instance of PCP.
⇒

By Lemma **??**, if (W,W') has a solution, then $\mathcal{L}(\mathcal{N}_{1-1^\infty}) \neq \emptyset$. By Lemma **??**, we know $\mathcal{L}(\mathcal{N}_0) = \emptyset$. Thus, if (W,W') has a solution, then $\mathcal{L}(\mathcal{N}_0) \neq \mathcal{L}(\mathcal{N}_{1-1^\infty})$.
⇐

By Lemma **??**, $\mathcal{L}(\mathcal{N}_0) = \emptyset$. If $\mathcal{L}(\mathcal{N}_0) \neq \mathcal{L}(\mathcal{N}_{1-1^\infty})$ then $\mathcal{L}(\mathcal{N}_{1-1^\infty}) \neq \emptyset$. So, by Lemma **??**, (W,W') has a solution. □

Theorem 1. *The Synchronizability Problem is undecidable for peer-to-peer systems.*

Proof. According to Lemma **??**, an instance of PCP has a solution if and only if the language of the peer-to-peer encoding of this instance is not synchronizable. The encoding of the Post Correspondence Problem is a reduction to the Synchronizability Problem for peer-to-peer systems. The Post Correspondence Problem is an undecidable problem, then the Synchronizability Problem for peer-to-peer systems is undecidable. □

Appendix B Proofs of Section ??

Lemma 4. *Let (W,W') an instance of PCP and $[\![\mathcal{W}, \mathcal{W}']\!]^{mail} = \mathcal{N}$ its encoding into communicating automata, then $\mathcal{L}(\mathcal{N}_0) = \emptyset$.*

Proof. By contradiction, we suppose $\mathcal{L}(\mathcal{N}_0) \neq \emptyset$. Thus let t a final send trace such that $t \in \mathcal{L}(\mathcal{N}_0)$. As $(q_e^W, q_e^{W'}, q_{ok}^L, q_{\$'}^I)$ is the unique final global state and by definition of final send trace, we have: $t = act_1 act_2 \ldots act_e$ such that

$$((q_0^W, q_0^{W'}, q_0^L, q_0^I), B^\emptyset) \xRightarrow[0]{act_1} C_1 \xRightarrow[0]{act_2} \ldots \xRightarrow[0]{act_e} ((q_e^W, q_e^{W'}, q_{ok}^L, q_{\$'}^I), B^\emptyset)$$

By construction, there is a unique transition in $\mathcal{B}_L$ to reach state q^L_{ok}: $q^L_{e'} \xrightarrow{!ok^{L \to I}}_L q^L_{ok}$, so $\exists act_k \in t$ such that $act_k = !ok^{L \to I}$. The communication

being synchronous, a sending must occur together with its reception (see rule (Sync. SEND)). By construction, there is no transition in $\mathcal{B}_I$ for the reception of message *ok*. So, it can not be send and $\mathcal{B}_L$ can not reach q^L_{ok}. Thus t does not exist and $\mathcal{L}(\mathcal{N}_0) = \emptyset$. □

Lemma 5. *For all instance (W,W') of PCP where* $[\![\mathcal{W}, \mathcal{W}']\!]^{mail} = \mathcal{N}$, *(W,W') has a solution if and only if* $\mathcal{L}(\mathcal{N}_{*-1^\infty}) \neq \emptyset$.

Proof. $\Rightarrow$
Let $\mathrm{Sol}_{(\mathrm{W,W'})} = (i_1, i_2, \ldots, i_m)$ be a solution of (W,W'). For each index $k \in \mathrm{Sol}_{(\mathrm{W,W'})}$, corresponding words in W and W' are $w_k = w_{k,1}w_{k,2} \ldots w_{k,|w_k|}$ and $w'_k = w'_{k,1}w'_{k,2} \ldots w'_{k,|w'_k|}$ respectively.

Take a send trace t^s of the following form:

$$\begin{aligned} t^s = {} & !i_1^{I\to W} \cdot !i_1'^{I\to W'} \cdot !i_2^{I\to W} \cdot !i_2'^{I\to W'} \cdot \ldots \cdot \\ & !i_m^{I\to W} \cdot !i_m'^{I\to W'} \cdot !\$^{I\to W} \cdot !\$'^{I\to W'} \cdot \\ & !x_1^{W\to L} \cdot !x_1'^{W'\to L} \cdot !x_2^{W\to L} \cdot !x_2'^{W'\to L} \cdot \ldots \cdot \\ & !x_y^{W\to L} \cdot !x_y'^{W'\to L} \cdot !end^{W\to L} \cdot !end'^{W'\to L} \cdot !ok^{L\to I} \end{aligned}$$

with

$$\begin{aligned} t^w &= (x_1^{W\to L}, x_2^{W\to L}, \ldots, x_y^{W\to L}, end^{W\to L}) \\ &= (w_{i_1,1}^{W\to L}, w_{i_1,2}^{W\to L}, \ldots, w_{i_1,|w_{i_1}|}^{W\to L}, w_{i_2,1}^{W\to L}, w_{i_2,2}^{W\to L}, \ldots, w_{i_2,|wi2|}^{W\to L}, \\ &\quad \ldots \\ &\quad w_{i_m,1}^{W\to L}, w_{i_m,2}^{W\to L}, \ldots, w_{i_m,|wi_m|}^{W\to L}) \end{aligned}$$

and

$$\begin{aligned} t^{w'} &= (x_1'^{W'\to L}, x_2'^{W'\to L}, \ldots, x_y'^{W'\to L}, end'^{W'\to L}) \\ &= (w_{i_1,1}'^{W'\to L}, w_{i_1,2}'^{W'\to L}, \ldots, w_{i_1,|w_{i_1}|'}'^{W'\to L}, w_{i_2,1}'^{W'\to L}, w_{i_2,2}'^{W'\to L}, \ldots, w_{i_2,|wi_2'|}'^{W'\to L}, \\ &\quad \ldots \\ &\quad w_{i_m,1}'^{W'\to L}, w_{i_m,2}'^{W'\to L}, \ldots, w_{i_m,|wi_m'|}'^{W'\to L}) \end{aligned}$$

We show that $t \in \mathcal{L}(\mathcal{N}_{*-1^\infty})$. Trace t corresponds to the following run: First, $\mathcal{B}_I$ sends indices to $\mathcal{B}_W$ and $\mathcal{B}_{W'}$.

$$((q_0^I, q_0^W, q_0^{W'}, q_0^L), B^\emptyset) \xrightarrow[*-1^\infty]{!i_1^{I\to W}} ((q_{i_1}^I, q_0^W, q_0^{W'}, q_0^L), B_1 = B^\emptyset\{b_W/b_W.i_1\})$$

$$\xrightarrow[*-1^\infty]{!i_1'^{I\to W'}} ((q_0^I, q_0^W, q_0^{W'}, q_0^L), B_1\{b_{W'}/b_{W'}.i_1'\})$$

$$\ldots$$

$$((q_0^I, q_0^W, q_0^{W'}, q_0^L), B_{1'}) \xrightarrow[*-1^\infty]{!i_m^{I\to W}} ((q_{i_m}^I, q_0^W, q_0^{W'}, q_0^L), B_{1''} = B_{1'}\{b_W/b_W.i_m\})$$

$$\xrightarrow[*-1^\infty]{!i_m'^{I\to W'}} ((q_0^I, q_0^W, q_0^{W'}, q_0^L), B_2 = B_{1''}\{b_{W'}/b_{W'}.i'_m\})$$

At this state, $\mathcal{B}_I$ sent all indices to $\mathcal{B}_W$ and $\mathcal{B}_{W'}$. He has left messages \$ and \$′ to send.

$$\xrightarrow[*-1^\infty]{!\$^{I\to W}} ((q_\$^I, q_0^W, q_0^{W'}, q_0^L), B_{2'} = B_2\{b_W/b_W.\$\})$$

$$\xrightarrow[*-1^\infty]{!\$'^{I\to W'}} ((q_{\$'}^I, q_0^W, q_0^{W'}, q_0^L), B_3 = B_{2'}\{b_{W'}/b_{W'}.\$'\})$$

At this state, buffers b_W and $b_{W'}$ are filled with indices sent by $\mathcal{B}_I$ and $\mathcal{B}_I$ is in its final state. Now, $\mathcal{B}_W$ and $\mathcal{B}_{W'}$ will read indices and send corresponding words to $\mathcal{B}_L$.

$$((q_{\$'}^I, q_0^W, q_0^{W'}, q_0^L), B_3) \xrightarrow[*-1^\infty]{?i_1^{I\to W}} ((q_{\$'}^I, q_{i_1,0}^W, q_0^{W'}, q_0^L), B_{3'} = B_3\{i_1.b_W/b_W\})$$

$$\xrightarrow[*-1^\infty]{?i_1'^{I\to W'}} ((q_{\$'}^I, q_{i_1,0}^W, q_{i_1,0}^{W'}, q_0^L), B_{3''} = B_{3'}\{i'_1.b_{W'}/b_{W'}\})$$

$$\xrightarrow[*-1^\infty]{!x_1^{W\to L}} ((q_{\$'}^I, q_{i_1,1}^W, q_{i_1,0}^{W'}, q_0^L), B_{3'''} = B_{3''}\{b_L/b_L.w_{i_1,1}\})$$

$$\xrightarrow[*-1^\infty]{!x_1'^{W'\to L}} ((q_{\$'}^I, q_{i_1,1}^W, q_{i_1,1}^{W'}, q_0^L), B_{3'''}\{b_L/b_L.w'_{i_1,1}\})$$

$$\ldots$$

$$((q_{\$'}^I, q_f^W, q_f^{W'}, q_0^L), B_4) \xrightarrow[*-1^\infty]{?\$^{I\to W}} ((q_{\$'}^I, q_\$^W, q_f^{W'}, q_0^L), B_{4'} = B_4\{\$ \cdot b_W/b_W\})$$

$$\xrightarrow[*-1^\infty]{?\$'^{I\to W'}} ((q_{\$'}^I, q_\$^W, q_\$^{W'}, q_0^L), B_{4''} = B_{4'}\{\$' \cdot b_{W'}/b_{W'}\})$$

$$\xrightarrow[*-1^\infty]{!end^{W\to L}} ((q_{\$'}^I, q_e^W, q_\$^{W'}, q_0^L), B_{4'''} = B_{4''}\{b_L/b_L.end\})$$

$$\xrightarrow[*-1^\infty]{!end'^{W'\to L}} ((q_{\$'}^I, q_e^W, q_e^{W'}, q_0^L), B_5 = B_{4'''}\{b_L/b_L.end'\})$$

At this state, $\mathcal{B}_W$ and $\mathcal{B}_{W'}$ have sent all letters to $\mathcal{B}_L$ and their buffers are empty. Now, $\mathcal{B}_L$ will compare these letters.

$$((q_{\$'}^I, q_e^W, q_e^{W'}, q_0^L), B_5) \xrightarrow[*-1^\infty]{?x_1^{W\to L}} ((q_{\$'}^I, q_e^W, q_e^{W'}, q_{x_1}^L), B_{5'} = B_5\{x_1.b_L/b_L\})$$

$$\xrightarrow[*-1^\infty]{?x_1'^{W'\to L}} ((q_{\$'}^I, q_e^W, q_e^{W'}, q_0^L), B_{5''} = B_{5'}\{x_1 \cdot b_L/b_L\})$$

$$\ldots$$

$$((q^I_{\$'}, q^W_e, q^{W'}_e, q^L_0), B_6) \xrightarrow[*-1^\infty]{?x_y^{W\to L}} ((q^I_{\$'}, q^W_e, q^{W'}_e, q^L_{x_y}), B_{6'} = B_6\{x_y \cdot b_L/b_L\})$$

$$\xrightarrow[*-1^\infty]{?x'^{W'\to L}_y} ((q^I_{\$'}, q^W_e, q^{W'}_e, q^L_0), B_{6''} = B_{6'}\{x'_y \cdot b_L/b_L\})$$
$$\xrightarrow[*-1^\infty]{?end^{W\to L}} ((q^I_{\$'}, q^W_e, q^{W'}_e, q^L_e), B_{6'''} = B_{6''}\{end \cdot b_L/b_L\})$$
$$\xrightarrow[*-1^\infty]{?end'^{W'\to L}} ((q^I_{\$'}, q^W_e, q^{W'}_e, q^L_{e'}), B_7 = B_{6'''}\{end' \cdot b_L/b_L\})$$

At this state, buffers b_L is empty and comparisons of letters have succeeded, as witnessed by its state $q^L_{e'}$. If it was not the case, it would be in state q^L_*. So, it can send message ok.

$$((q^I_{\$'}, q^W_e, q^{W'}_e, q^L_{e'}), B_7) \xrightarrow[*-1^\infty]{!ok^{L\to I}} ((q^I_{\$'}, q^W_e, q^{W'}_e, q^L_{ok}), B_7\{b_I/b_I.ok\})$$

Global state $(q^I_{\$'}, q^W_e, q^{W'}_e, q^L_{ok})$ reached with this run is the final global state of the network and $t \in \mathcal{L}(\mathcal{N}_{*-1^\infty})$. Thus $\mathcal{L}(\mathcal{N}_{*-1^\infty}) \neq \emptyset$, concluding this side of the proof.

$\Leftarrow$

If $\mathcal{L}(\mathcal{N}_{*-1^\infty}) \neq \emptyset$ then $\exists\ t \in \mathcal{L}(\mathcal{N}_{*-1^\infty})$. As $(q^I_{\$'}, q^W_e, q^{W'}_e, q^L_{ok})$ is the unique final global state and by definition of language, t is a final send trace and we have: $t = act_1 act_2 \dots act_e$ such that

$$((q^I_0, q^W_0, q^{W'}_0, q^L_0), B^\emptyset) \xrightarrow[0]{act_1} C_1 \xrightarrow[0]{act_2} \dots \xrightarrow[0]{act_e} ((q^I_{\$'}, q^W_e, q^{W'}_e, q^L_{ok}), B_e)$$

Notice that in order to reach $q^I_\$$:

- $\exists k \in \mathbb{N}^+$ such that t contains $m^{I\to W}_1,\ m^{I\to W}_2, \dots,\ m^{I\to W}_k$ in this order and $m^{I\to W'}_{1'},\ m^{I\to W'}_{2'}, \dots,\ m^{I\to W'}_{k'}$ in this order,
- $\forall i, i' \in [1,k]$, $m^{W\to I}_i = j^{W\to I}$ and $m^{W'\to I}_{i'} = j'^{W'\to I}$, with $j \in [1,n]$, i.e., sequences of indices sent by $\mathcal{B}_I$ to $\mathcal{B}_W$ and $\mathcal{B}_{W'}$ are equal.

Moreover, to reach q^W_e and $q^{W'}_e$, we have that:

- $\exists\ m_i, m_{i'} \in t,\ i, i' \in [1,e]$, where $m_i = ?j^{I\to W}$ and $m_{i'} = ?j'^{I\to W'}$, $j \in [1,n]$, i.e., $\mathcal{B}_W$ and $\mathcal{B}_{W'}$ receive at least one index (and send the associated word),
- $\exists\ m_f, m_{f'} \in t,\ f, f' \in [1,e],\ i < f,\ i' < f', f < f'$, where $m_f = !end^{W\to L}, m_{f'} = !end'^{W'\to L}$, i.e., $\mathcal{B}_W$ and $\mathcal{B}_{W'}$ send messages of end of sequence of letters to $\mathcal{B}_L$, after receiving at least one index.

Finally, to reach q^L_{ok}, we deduce that:

- $\exists l \in \mathbb{N}^+$ such that t contains $m_1^{W\to L},\ m_2^{W\to L}, \ldots,\ m_l^{W\to L}$ in this order and $m_1'^{W'\to L},\ m_2'^{W'\to L}, \ldots,\ m_l'^{W'\to L}$ in this order,
- $\forall i \in [1, l]$, $m_i^{W\to I} = \alpha^{W\to I}$ and $m_i'^{W'\to L} = \alpha'^{W'\to L}$, with $\alpha \in \Sigma$, i.e., sequences of letters sent by $\mathcal{A}_W$ and $\mathcal{A}_{W'}$ to $\mathcal{A}_L$ must be equal.
- $\exists act_z \in t$ such that $act_z = ok^{L\to I}$ and $\mathcal{A}_L$ can reach q_{ok}^L.

Then, if the network is in a final global state, sequences of letters sent by $\mathcal{A}_W$ and $\mathcal{A}_{W'}$ are equal. So trace t contains a solution of instance (W,W') that can be found by extracting the messages sent by $\mathcal{B}_I$ to $\mathcal{B}_W$:

$$Sol_{(\mathrm{W,W'})} = (m_1^{I\to W}, m_2^{I\to W}, \ldots, m_k^{I\to W})$$

□

Lemma 6. *For all instance (W,W') of PCP where* $[\![\mathcal{W}, \mathcal{W}']\!]^{mail} = \mathcal{N}$*, (W,W') has a solution if and only if* $\mathcal{L}(\mathcal{N}_0) \neq \mathcal{L}(\mathcal{N}_{*-1^\infty})$.

Proof. Let (W,W') be an instance of PCP.
⇒

By Lemma **??**, if (W,W') has a solution, then $\mathcal{L}(\mathcal{N}_{*-1^\infty}) \neq \emptyset$. By Lemma **??**, we know $\mathcal{L}(\mathcal{N}_0) = \emptyset$. Thus, if (W,W') has a solution, then $\mathcal{L}(\mathcal{N}_0) \neq \mathcal{L}(\mathcal{N}_{*-1^\infty})$.
⇐

By Lemma **??**, $\mathcal{L}(\mathcal{N}_0) = \emptyset$. If $\mathcal{L}(\mathcal{N}_0) \neq \mathcal{L}(\mathcal{N}_{*-1^\infty})$ then $\mathcal{L}(\mathcal{N}_{*-1^\infty}) \neq \emptyset$. So, by Lemma **??**, (W,W') has a solution. □

Theorem 2. *The Synchronizability Problem is undecidable for mailbox systems.*

Proof. According to Lemma **??**, an instance of PCP has a solution if and only if the language of the mailbox encoding of this instance is not synchronizable. The encoding of the Post Correspondence Problem is a reduction to the Synchronizability Problem for mailbox systems. The Post Correspondence Problem is an undecidable problem, then the Synchronizability Problem for mailbox systems is undecidable. □

Bibliography

[1] L. Akroun and G. Salaün. Automated verification of automata communicating via FIFO and bag buffers. *Formal Methods in System Design*, 52(3):260–276, June 2018.

[2] L. Akroun, G. Salaün, and L. Ye. Automated Analysis of Asynchronously Communicating Systems. In D. Bošnački and A. Wijs, editors, *Model Checking Software*, volume 9641, pages 1–18. Springer International Publishing, Cham, 2016.

[3] S. Basu and T. Bultan. On deciding synchronizability for asynchronously communicating systems. *Theoretical Computer Science*, 656:60–75, Dec. 2016.

[4] A. Bouajjani, C. Enea, K. Ji, and S. Qadeer. On the Completeness of Verifying Message Passing Programs under Bounded Asynchrony. *arXiv:1804.06612 [cs]*, Apr. 2018. arXiv: 1804.06612.

[5] A. Finkel and E. Lozes. Synchronizability of Communicating Finite State Machines is not Decidable. *arXiv:1702.07213 [cs]*, Feb. 2017. arXiv: 1702.07213.

[6] H. GARAVEL. Introduction au langage LOTOS. page 14, 1990.

[7] B. Genest, D. Kuske, and A. Muscholl. On communicating automata with bounded channels. page 21, 2007.

[8] D. Kuske and A. Muscholl. Communication automata. 2010.

[9] R. Mateescu. Specification and Analysis of Asynchronous Systems using CADP. In S. Merz and N. Navet, editors, *Modeling and Verification of Real-Time Systems*, pages 141–169. ISTE, London, UK, Jan. 2008.

[10] A. Mazurkiewicz. Concurrent Program Schemes and their Interpretations. *DAIMI Report Series*, 6(78), July 1977.

[11] R. Milner. *Communication and Concurrency. International series in computer science.* Prentice hall Englewood Cliffs, 1989.

[12] E. L. Post. A variant of a recursively unsolvable problem. *Bulletin of the American Mathematical Society*, 52(4):264–268, 1946.

[13] M. Sipser. *Introduction to the theory of computation*. Course Technology Cengage Learning, Boston, MA, 3rd ed edition, 2012.