# Wait-Only Broadcast Protocols are Easier to Verify

**Lucie Guillou**
IRIF, CNRS, Université Paris Cité, France

**Arnaud Sangnier**
DIBRIS, Università di Genova, Italy

**Nathalie Sznajder**
LIP6, CNRS, Sorbonne Université, France

## Abstract

We study networks of processes that all execute the same finite-state protocol and communicate via broadcasts. We are interested in two problems with a parameterized number of processes: the synchronization problem which asks whether there is an execution which puts all processes on a given state; and the repeated coverability problem which asks if there is an infinite execution where a given transition is taken infinitely often. Since both problems are undecidable in the general case, we investigate those problems when the protocol is *Wait-Only*, i.e., it has no state from which a process can both broadcast and receive messages. We establish that the synchronization problem becomes Ackermann-complete, and the repeated coverability problem is in ExpSpace and PSpace-hard.

**2012 ACM Subject Classification** Theory of computation → Formal languages and automata theory

**Keywords and phrases** Parameterised verification, Reachability, Broadcast

**Digital Object Identifier** 10.4230/LIPIcs.MFCS.2025.42

**Funding** ANR project PaVeDyS (ANR-23-CE48-0005)


50th International Symposium on Mathematical Foundations of Computer Science (MFCS 2025).
Editors: Paweł Gawrychowski, Filip Mazowiecki, and Michał Skrzypczak; Article No. 42; pp. 42:1–42:32
Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

# 1 Introduction

Distributed systems are at the core of modern computing, and are widely used in critical applications such as sensor networks, or distributed databases. These systems rely on protocols to enable communication, maintain coherence and coordination between the different processes. Because of their distributed nature, such protocols have proved to be error-prone. As a result, the formal verification of distributed systems has become an essential area of research. Formal verification of distributed systems presents unique challenges compared to fthe one of centralized systems. One of the most significant issue is the state explosion problem: the behavior of multiple processes that execute concurrently and exchange information often lead to state spaces that grow exponentially with the number of processes, making the analysis highly challenging. When the systems are parameterized, meaning designed to operate for an arbitrary number of processes, which is often the case in distributed protocols, classical techniques are not useful anymore. The difficulty shifts from state explosion to dealing with an infinite number of system configurations, which leads to undecidability in the general case [1]. Despite these challenges, verification becomes more tractable in certain restricted settings. For instance, in parameterized systems, the behavior of individual processes needs not always to be explicitly represented, which can mitigate state explosion. This has motivated researchers to focus on specific subclasses of systems or communication models where decidability can be achieved without sacrificing too much expressiveness. One such restriction involves protocols where processes are anonymous (i.e., without identities), and communicate via simple synchronous mechanisms, such as rendezvous [9], where two processes exchange a message synchronously. Several variants of this model have been studied, such as communication via a shared register containing some value from a finite set [7], or non-blocking rendezvous [10, 4], where a sender is not prevented from sending a message when no process is ready to receive it. In all these cases, the processes execute the same protocol, which is described by a finite automaton.

A more expressive model is broadcast protocols, introduced by Emerson and Namjoshi [6]. In these protocols, a process can send a broadcast message that is received simultaneously by all other processes (or by all neighboring processes in a network). Several key verification problems arise in this context, including the coverability problem, which asks whether a process can eventually reach a specific (bad) state starting from an initial configuration. Another important property is synchronization, which asks whether all processes can simultaneously reach a given state. Liveness properties, like determining whether infinite executions exist, ensure that certain behaviors occur indefinitely. While broadcast protocols allow more powerful communication than rendezvous protocols, this added expressiveness comes at the cost of increased verification complexity. Indeed, rendezvous communication can be simulated using broadcasts [8]. However, verification becomes significantly harder: while coverability and synchronization can be solved in polynomial time for rendezvous protocols [9], they become respectively Ackermann-complete [8, 20] and undecidable (we show it in this paper) for broadcast protocols. Liveness properties are also undecidable for broadcast protocols [8]. The challenge in verifying broadcast protocols can be understood through their relationship with Vector Addition Systems with States (VASS). Rendezvous-based protocols can be encoded in a VASS, where counters track the number of processes in each state. A rendezvous is simulated by decreasing the counters corresponding to the sender and receiver states and increasing the counters for their post-communication states. While using a VASS for protocols using rendezvous is certainly not the way to go due to the complexity gap between problems on VASS and the existing known polynomial time algorithms to solve

verification problems, it is interesting to note that this encoding fails for broadcast protocols because a broadcast message must be received by all processes in a given state, requiring a bulk transfer of a counter value—something not expressible in classical VASS without adding powerful operations such as transfer arcs, which leads to an undecidable reachability problem [21]. However, in some cases, verification of broadcast protocols can become easier, complexity-wise. One example is the setting of Reconfigurable Broadcast Networks, in which communication between processes can be lossy [5]. Liveness properties become decidable in that case [5] and even polynomial time [2].

Another such case is given by a syntactic restriction known as *Wait-Only protocols*, introduced in [10] in the context of non-blocking rendezvous communication. In Wait-Only protocols, a process cannot both send and receive a message from the same state. From a practical perspective, Wait-Only protocols were proposed to model the non-blocking rendezvous semantics used in Java's `wait/notify/notifyAll` synchronization mechanism and C's `wait/signal/broadcast` operations with conditional variables, commonly found in multi-threaded programs. In both languages, threads can be awakened by the reception of a message. This naturally leads to distinguishing between action and waiting states: a sleeping thread cannot act on its own and can only be awakened by a signal. This restriction simplifies the possible behaviours of the system, potentially reducing the complexity of the verification. Actually, the coverability problem for Wait-Only broadcast protocols becomes PSpace-complete [11]. Indeed, when processes are in a state where they can receive broadcasts, they cannot send messages, meaning they will move together to the next state, reducing the need for precise counting. Moreover, when a process is in a broadcasting state, it remains there until it decides to send a message, simplifying execution reconstruction and enabling better verification algorithms. By leveraging these properties, we design algorithms for the synchronization problem and liveness properties, obtaining new decidability results for these challenging problems. Proofs omitted due to space contraints can be found in [12].

## 2 Model and Verification Problems

In this section, we provide the description of a model for networks with broadcast communication together with some associated verification problems we are interested in. First we introduce some useful mathematical notations. We use $\mathbb{N}$ to represent the set of natural numbers, $\mathbb{N}_{>0}$ to represent $\mathbb{N} \setminus \{0\}$, and for $n, m \in \mathbb{N}$ with $n \leq m$, we denote by $[n, m]$ the set $\{n, n+1, \ldots, m\}$. For a finite set $E$ and a natural $n > 0$, the set $E^n$ represents the $n$-dimensional vectors with values in $E$ and for $v \in E^n$ and $1 \leq i \leq n$, we use $v(i)$ to represent the $i$-th value of $v$. Furthermore, for such a vector $v$, we denote by $||v||$ its dimension $n$.

**Networks of Processes using Broadcast Communication.** In the networks we consider, all the processes execute the same protocol, given by a finite state machine. The actions of this machine can either broadcast a message $a$ to the other processes (denoted by $!!a$) or receive a message $a$ (denoted by $?a$). For a finite alphabet of messages $\Sigma$, we use the following notations: $!!\Sigma \stackrel{\text{def}}{=} \{!!a \mid a \in \Sigma\}$, $?\Sigma \stackrel{\text{def}}{=} \{?a \mid a \in \Sigma\}$ and $\mathsf{Op}_\Sigma \stackrel{\text{def}}{=} !!\Sigma \cup ?\Sigma$.

▶ **Definition 2.1.** *A* protocol $P$ *is a tuple* $(Q, \Sigma, q_{in}, T)$ *where* $Q$ *is a finite set of states,* $\Sigma$ *is a finite set of messages,* $q_{in} \in Q$ *is an initial state, and* $T \subseteq Q \times \mathsf{Op}_\Sigma \times Q$ *is a set of transitions.*

For all $q \in Q$, we define $R(q)$ as the set of messages that can be received from state $q$, given by $\{a \in \Sigma \mid \exists q' \in Q, (q, ?a, q') \in T\}$. A protocol $P = (Q, \Sigma, q_{in}, T)$ is *Wait-Only* whenever for all $q \in Q$, either $\{q' \in Q \mid (q, \alpha, q') \in T \text{ with } \alpha \in ?\Sigma\} = \emptyset$, or $\{q' \in$

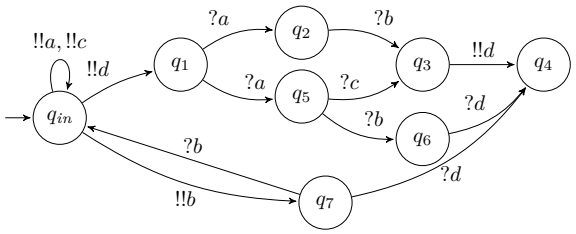


**Figure 1** Example of a Wait-Only protocol $P$.

$Q \mid (q, \alpha, q') \in T$ with $\alpha \in !!\Sigma\} = \emptyset$. In a Wait-Only protocol, a state $q \in Q$ such that $\{q' \in Q \mid (q, \alpha, q') \in T$ with $\alpha \in ?\Sigma\} \neq \emptyset$ is called a *waiting state*. A state that is not a waiting state is an *action state*. We denote by $Q_W$ the set of waiting states and by $Q_A$ the set of action states (hence $Q_A = Q \setminus Q_W$). Observe that if $q_{in} \in Q_W$, then no process is ever able to broadcast messages, for this reason we will always assume that $q_{in} \in Q_A$.

For $n \in \mathbb{N}_{>0}$, an $n$-process *configuration* of a protocol $P = (Q, \Sigma, q_{in}, T)$ is a vector $C \in Q^n$ where $C(e)$ represents the state of the $e$-th process in $C$ for $1 \leq e \leq n$. The configuration is said to be *initial* whenever $C(e) = q_{in}$ for all $1 \leq e \leq n$. The dimension $||C||$ hence characterizes the number of processes in $C$. For a state $q \in Q$, we use $C^{-1}(q)$ to represent the set of processes in state $q$, formally $C^{-1}(q) = \{1 \leq e \leq ||C|| \mid C(e) = q\}$. For a subset $A \subseteq [1, ||C||]$ of processes, we use $C(A)$ to identify the set of states of processes in $A$, formally $C(A) = \{C(e) \mid e \in A\}$. We let $\mathcal{C}$ [resp. $\mathcal{I}$] be the set of all configurations [resp. initial configurations] of $P$.

We now define the broadcast network semantics associated to a protocol $P = (Q, \Sigma, q_{in}, T)$. For configurations $C, C' \in \mathcal{C}$, transition $t = (q, !!a, q') \in T$ and $e \in [1, ||C||]$, we write $C \xrightarrow{e,t} C'$ whenever $||C|| = ||C'||$, $C(e) = q$, $C'(e) = q'$ and, for all $e' \in [1, ||C||] \setminus \{e\}$, either a reception occurs, i.e., $(C(e'), ?a, C'(e')) \in T$, or the process cannot receive $a$, in which case $(a \notin R(C(e'))$ and $C(e') = C'(e'))$. Intuitively, the $e$-th process of $C$ broadcasts a message $a$, which is received by all processes able to receive it, while the other processes remain in their current state. We note $C \to C'$ if there exists $e \in [1, ||C||]$ and $t \in T$ such that $C \xrightarrow{e,t} C'$ and use $\to^*$ [resp. $\to^+$] for the reflexive and transitive [resp. transitive] closure of $\to$. A finite [resp. infinite] execution is a finite [resp. infinite] sequence of configurations $\rho = C_0 \dots C_\ell$ [resp. $\rho = C_0 \dots$] such that $C_0 \in \mathcal{I}$ and $C_{i-1} \to C_i$ for all $0 < i \leq \ell$ [resp. for all $i > 0$]. Its length $|\rho|$ is equal to $\ell + 1$ [resp. $\omega$]. We write $\Lambda$ [resp. $\Lambda_\omega$] for the set of finite [resp. infinite] executions. For an execution $\rho = C_0 C_1 \dots \in \Lambda \cup \Lambda_\omega$ and $0 \leq i < |\rho|$, we use $\rho_i$ to denote $C_i$, the $i$-th configuration. We use $\#\mathbf{proc}(\rho)$ to represent $||C_0||$, the number of processes in the execution. For $0 \leq i < j < |\rho|$, we define $\rho_{i,j} \stackrel{\text{def}}{=} \rho_i \dots \rho_j$. We also let $\rho_{\geq i} \stackrel{\text{def}}{=} \rho_i \rho_{i+1} \dots$ and $\rho_{\leq i} \stackrel{\text{def}}{=} \rho_0 \dots \rho_i$. For $e \in [1, \#\mathbf{proc}(\rho)]$, we denote by $\rho(e)$ the sequence of states $\rho_0(e)\rho_1(e) \dots$.

▶ **Example 2.2.** Figure 1 illustrates an example of a Wait-Only protocol. The waiting states are $q_1, q_2, q_5, q_6$, and $q_7$, with $R(q_5) = \{b, c\}$. Here is one of its execution with 3 processes:

$$(q_{in}, q_{in}, q_{in}) \xrightarrow{1,(q_{in},!!d,q_1)} (q_1, q_{in}, q_{in}) \xrightarrow{2,(q_{in},!!d,q_1)} (q_1, q_1, q_{in}) \xrightarrow{3,(q_{in},!!a,q_{in})} (q_5, q_2, q_{in})$$
$$\xrightarrow{3,(q_{in},!!c,q_{in})} (q_3, q_2, q_{in}) \xrightarrow{3,(q_{in},!!b,q_7)} (q_3, q_3, q_7).$$

**Verification Problems.** We investigate two verification problems over broadcast networks: the synchronization problem (Synchro) and the repeated coverability problem (RepCover).

SYNCHRO asks, given a protocol $P = (Q, \Sigma, q_{in}, T)$ and a state $q_f \in Q$, whether there exist $C \in \mathcal{I}$ and $C' \in \mathcal{C}$ such that $C \to^* C'$, and $C'(e) = q_f$ for all $e \in [1, ||C'||]$. REPCOVER asks, given a protocol $P = (Q, \Sigma, q_{in}, T)$ and a transition $t = (q, !!a, q') \in T$, whether there exist $\rho \in \Lambda_\omega$ and $e \in [1, \#\mathbf{proc}(\rho)]$, such that for all $i \in \mathbb{N}$, there exists $j > i$ verifying $\rho_j \xrightarrow{e,t} \rho_{j+1}$.

▶ **Theorem 2.3.** *SYNCHRO and REPCOVER are undecidable.*

We give the proof of the undecidability of SYNCHRO in [12], while the undecidability of REPCOVER was established in [8, Theorem 5.1]. In the remainder of the paper, we show that by restricting to Wait-Only protocols, one can regain decidability.

▶ Remark 2.4. Throughout the remainder of this paper, we will consider protocols without self-loop broadcast transitions of the form $(q, !!a, q)$. Such transitions can be transformed into two transitions $(q, !!a, p_q)$, $(p_q, !!\$, q)$ where $p_q$ is a new state added to the set of states and \$ is a new message added to the alphabet. This transformation of the protocol is equivalent to the original one with respect to SYNCHRO and REPCOVER.

**Vector Addition System with States.** In the following, we will extensively rely on the model of Vector Addition Systems with States (VASS) which we introduce below. A VASS is a tuple $\mathcal{V} = (\mathsf{Loc}, \ell_0, \mathsf{X}, \Delta)$ where $\mathsf{Loc}$ is a finite set of locations, $\ell_0 \in \mathsf{Loc}$ is the *initial* location, $\mathsf{X}$ is a finite set of natural variables, called counters, and $\Delta \subseteq \mathsf{Loc} \times \mathbb{Z}^{\mathsf{X}} \times \mathsf{Loc}$ is a finite set of transitions. A configuration of $\mathcal{V}$ is a pair $(\ell, v)$ in $\mathsf{Loc} \times \mathbb{N}^{\mathsf{X}}$ where $v$ provides a value for each counter. The initial configuration is $(\ell_0, \mathbf{0})$ where $\mathbf{0}(\mathsf{x}) = 0$ for all $\mathsf{x} \in \mathsf{X}$. Given two configurations $(\ell, v)$, $(\ell', v')$ and a transition $(\ell, \delta, \ell') \in \Delta$, we write $(\ell, v) \overset{\delta}{\rightsquigarrow} (\ell', v')$ whenever $v'(\mathsf{x}) = v(\mathsf{x}) + \delta(\mathsf{x})$ for all counters $\mathsf{x} \in \mathsf{X}$. We simply use $(\ell, v) \rightsquigarrow (\ell', v')$ when there exists $(\ell, \delta, \ell') \in \Delta$ such that $(\ell, v) \overset{\delta}{\rightsquigarrow} (\ell', v')$. We denote $\rightsquigarrow^*$ for the transitive and reflexive closure of $\rightsquigarrow$. An (infinite) execution (or run) of the VASS is a (infinite) sequence of configurations $(\ell_0, v_0), (\ell_1, v_1), \ldots, (\ell_n, v_n)$, starting with the initial configuration and such that $(\ell_i, v_i) \rightsquigarrow (\ell_{i+1}, v_{i+1})$ for all $0 \leq i < n$.

REACH, the well-known reachability problem for VASS, is defined as follows: given a VASS $\mathcal{V} = (\mathsf{Loc}, \ell_0, \mathsf{X}, \Delta)$ and a location $\ell_f \in \mathsf{Loc}$, is there an execution from $(\ell_0, \mathbf{0})$ to $(\ell_f, \mathbf{0})$?

▶ **Theorem 2.5** (Membership [18], Hardness [3, 17])**.** *REACH is Ackermann-complete.*

## 3 Solving Synchro for Wait-Only Protocols

To solve SYNCHRO we show in this section that we can build a VASS that simulates the behavior of a broadcast network running a Wait-Only protocol. An intuitive way to proceed, inspired by the counting abstraction proposed for protocols communicating by pairwise rendez-vous communication [9], consists in having one counter per state of the protocol, that stores the number of processes that populate that state. Initially, the counter associated with the initial state can be incremented as much as one wants: this will determine the number of processes of the execution simulated in the VASS. Then, the simulation of broadcast messages $(q, !!a, q')$ amounts to decrementing the counter associated to $q$ and increment the counter associated to $q'$. However, simulating receptions of the message (e.g. a transition $(p, ?a, p')$), requires to empty the counter associated to $p$ and transfer its value to the counter associated to $q'$. This transfer operation is not something that can be done in VASS unless the model is extended with special transfer transitions, leading to an undecidable reachability problem [21].

To circumvent this problem, we rely on two properties of Wait-Only protocols: (1) processes that occur to be in the same waiting state follow the same path of reception in the protocol, and end in the same action state (we show how to take care of potential non-determinism in the next section) and (2) processes in an action state cannot be influenced by the behaviour of other processes as they cannot receive any message. Thanks to property (1), instead of precisely tracking the number of processes in each waiting state, we only count the processes in a "waiting region" – a connected component of waiting states populated by processes that will all reach the same action state simultaneously. The waiting region allows us also to monitor when the processes go out from a waiting region to an action state. We will explain later how we use property (2) to handle the transfer of values of counters within the VASS semantics, and thus simulating processes leaving a waiting region.

For the rest of this section, we fix $P = (Q, \Sigma, q_{in}, T)$ a Wait-Only protocol (with $Q_W$ [resp. $Q_A$] the set of waiting states [resp. action states]) for which we assume w.l.o.g. the existence of an *uncoverable state* $q_u \in Q$ verifying $q \neq q_u$ and $q' \neq q_u$ for all $(q, \alpha, q') \in T$ and $q_u \neq q_{in}$. Furthermore, we consider a final waiting state $q_f \in Q_W$. To ease the reading, we assume in this section that the final state $q_f$ is a waiting state, but our construction could be adapted to the case where $q_f$ is an action state. Moreover, we show in the next section that Synchro in this latter case is easier to solve.

## 3.1 Preliminary properties

We present here properties satisfied by Wait-Only protocols, which we will rely on for our construction. We first show with Lemma 3.1 that if, during an execution, two processes fulfill two conditions: $(i)$ they are on the same waiting state $q_w$, and $(ii)$ the next action state they reach (not necessarily at the same time) is the same (namely $q_a$), then one can build an execution where they both follow the same path between $q_w$ and $q_a$. This is trivial for *deterministic* protocols (where for all $q \in Q$ and $\alpha \in \mathsf{Op}_\Sigma$, there is at most one transition of the form $(q, \alpha, q') \in T$) and is also true for non-deterministic protocols.

For an execution $\rho \in \Lambda \cup \Lambda_\omega$ of $P$, an index $0 \leq j < |\rho|$, a waiting state $q \in Q_W$ and a process number $e \in [1, \#\mathbf{proc}(\rho)]$ such that $\rho_j(e) = q$, we define $\mathbf{na\text{-}index}(\rho, j, e) = \min\{k \mid j \leq k \leq |\rho| \text{ and } \rho_k(e) \in Q_A\}$, i.e. the first moment after $\rho_j$ where the process $e$ reaches an action state (if such moment does not exist, it is set to $|\rho|$). We note $\mathbf{na\text{-}state}(\rho, j, e) \stackrel{\text{def}}{=} \rho_{\mathbf{na\text{-}index}(\rho,j,e)}(e)$ the next action state for process $e$ from $\rho_j$ if $\mathbf{na\text{-}index}(\rho, j, e) \neq |\rho|$ and otherwise we take the convention that $\mathbf{na\text{-}state}(\rho, j, e)$ is equal to $q_u$, the uncoverable state of the protocol $P$. Finally, we let $\mathbf{na}(\rho, j, e) = (\mathbf{na\text{-}state}(\rho, j, e), \mathbf{na\text{-}index}(\rho, j, e))$.

▶ **Lemma 3.1.** *Let $\rho \in \Lambda \cup \Lambda_\omega$ be an execution of $P$. If there exist $e_1, e_2 \in [1, \#\mathbf{proc}(\rho)]$ and $0 \leq j < |\rho|$ such that (i) $\rho_j(e_1) = \rho_j(e_2) \in Q_W$, (ii) $\mathbf{na}(\rho, j, e_1) = (q, j_1)$, and (iii) $\mathbf{na}(\rho, j, e_2) = (q, j_2)$ with $j_1 \leq j_2 < |\rho|$, then there exists an execution $\rho'$ of the form $\rho_{\leq j}\rho'_{j+1} \dots \rho'_{j_2}\rho_{\geq j_2+1}$ such that $\rho'_k(e_1) = \rho'_k(e_2) = \rho_k(e_1)$ for all $j+1 \leq k \leq j_1$, and $\rho'_k(e_1) = \rho_k(e_1)$ and $\rho'_k(e_2) = q$ for all $j_1 < k \leq j_2$. In particular $\mathbf{na}(\rho', j, e_1) = \mathbf{na}(\rho', j, e_2) = (q, j_1)$.*

▶ **Example 3.2.** Consider the protocol $P$ in Figure 1 and the execution $\rho$ of Example 2.2. We have $\rho_2(1) = \rho_2(2) = q_1$ and $\mathbf{na}(\rho, 2, 1) = (q_3, 4)$ and $\mathbf{na}(\rho, 2, 2) = (q_3, 5)$. Applying Lemma 3.1, we build: $\rho' = (q_{in}, q_{in}, q_{in}) \to (q_1, q_{in}, q_{in}) \to (q_1, q_1, q_{in}) \to (q_5, q_5, q_{in}) \to (q_3, q_3, q_{in}) \to (q_3, q_3, q_7)$ where process 1 and process 2 follow the same path between $q_1$ and $q_3$. However, consider the following events from $(q_1, q_1, q_{in})$: $(q_1, q_1, q_{in}) \to (q_2, q_5, q_{in}) \to (q_3, q_6, q_7) \to (q_4, q_4, q_4)$, here we cannot apply the lemma as from $q_1$, processes 1 and 2 reach two different next action states ($q_3$ and $q_4$).

The above lemma enables us to focus on a specific subset of executions for SYNCHRO, which we refer to as well-formed. In these executions, at any moment, two processes in the same waiting state and with the same next action state, follow the same path of receptions. Formally, an execution $\rho$ of a protocol $P$ is *well-formed* iff for all $0 \leq i < |\rho|$, for all $e_1, e_2 \in [1, \#\mathbf{proc}(\rho)]$ such that $\rho_i(e_1) = \rho_i(e_2) \in Q_W$ and $\mathbf{na\text{-}state}(\rho, i, e_1) = \mathbf{na\text{-}state}(\rho, i, e_2) = q$, it holds that $\rho_k(e_1) = \rho_k(e_2)$ for all $i \leq k \leq \mathbf{na\text{-}index}(\rho, i, e_1)$. From Lemma 3.1, we immediately get:

▶ **Corollary 3.3.** *There exists an execution $\rho = C_0 \dots C_n$ such that $C_n(e) = q_f$ for all $e \in [1, \#\mathbf{proc}(\rho)]$ iff there exists a well-formed execution from $C_0$ to $C_n$.*

We finally provide a result which will allow us to bound the size of the VASS that we will build from the given Wait-Only protocol. Given an execution $\rho \in \Lambda \cup \Lambda_\omega$ of $P$, an index $0 \leq j < |\rho|$ and an action state $q_a \in Q_A$, we define $\mathbf{na\text{-}index\text{-}set}(\rho, j, q_a)$ as the set of indices where processes in a waiting state at $\rho_j$ will reach $q_a$, if it is their next action state: $\mathbf{na\text{-}index\text{-}set}(\rho, j, q_a) \stackrel{\text{def}}{=} \{i \mid \text{ there exists } e \in [1, \#\mathbf{proc}(\rho)] \text{ s.t. } \rho_j(e) \in Q_W \text{ and } \mathbf{na}(\rho, j, e) = (q_a, i)\}$.

▶ **Lemma 3.4.** *For all well-formed executions $\rho$, for all $0 \leq j < |\rho|$, and for all $q_a \in Q_A$, we have $|\mathbf{na\text{-}index\text{-}set}(\rho, j, q_a)| \leq |Q_W|$.*

**Proof.** If we have $|\mathbf{na\text{-}index\text{-}set}(\rho, j, q_a)| > |Q_W|$, by pigeon hole principle there exist $e_1, e_2 \in [1, \#\mathbf{proc}(\rho)]$ such that $\rho_j(e_1) = \rho_j(e_2) \in Q_W$ and such that $\mathbf{na}(\rho, j, e_1) = (q_a, i_1)$ and $\mathbf{na}(\rho, j, e_2) = (q_a, i_2)$ with $i_1 < i_2$. Consequently $\rho_{i_1}(e_1) = q_a \in Q_A$ and $\rho_{i_1}(e_2) \in Q_W$ hence $\rho_{i_1}(e_1) \neq \rho_{i_1}(e_2)$, which contradicts the definition of well-formedness. ◀

## 3.2 Building the VASS that Simulates a Broadcast Network

**Summaries.** We present here the formal tool we use to represent processes in waiting states. We begin by introducing some notations. A *print* $\mathsf{pr}$ is a set of waiting states. Given a non empty subset of states $A = \{q_1, \dots, q_n\} \subseteq Q$, we define a configuration $\overset{\bullet}{A} \in Q^n$ such that $\overset{\bullet}{A}(e) = q_e$ for all $e \in [1, n]$. When $A = \{q\}$, we write $\overset{\bullet}{q} \in Q^1$ instead of $\{\overset{\bullet}{q}\}$. Conversely, given a configuration $C$, we define $\mathbf{set}(C) = \{q \in Q \mid C^{-1}(q) \neq \emptyset\}$. For two configurations $C \in Q^n$ and $C' \in Q^m$, we let $C \oplus C'$ be the configuration $C'' \in Q^{n+m}$ defined by: $C''(e) = C(e)$ if $1 \leq e \leq n$ and $C''(e) = C'(e - n)$ if $n + 1 \leq e \leq m + n$. A *summary* $S$ is a tuple $(\mathsf{pr}, q_a, k)$ such that $\mathsf{pr} \subseteq Q_W$ is a print, $q_a \in Q_A$ is an *exit state*, and $k \in [1, |Q_W| + 1]$ is an identifier. The label of $S$ is $\mathbf{lab}(S) = (q_a, k)$ and we denote its print by $\mathbf{print}(S)$. Finally, we define a *special* summary `Done`.

In our construction, each process in a waiting state is associated with a summary while processes in action states are handled by the counters of the VASS. Intuitively, a summary $(\mathsf{pr}, q_a, k)$ represents a set of processes lying in the set $\mathsf{pr} \subseteq Q_W$ that will reach the same next action state $q_a$ simultaneously (this restriction is justified by Lemma 3.1). Since the set of waiting states $\mathsf{pr}$ evolve during the simulation, each summary must be uniquely identified. To achieve this, we use an integer $k \in [1, |Q_W| + 1]$. Hence each summary is identified with the pair $(q_a, k)$. We do not need more than $|Q_W| + 1$ different identifiers, because when a process enters a new summary (i.e it arrives in a waiting state $q_w$ from an action state), aiming for next action state $q_a$, it either joins an existing summary with exit state $q_a$, or create a new one. In the latter case, well-formed executions ensure that no existing summary $S = (\mathsf{pr}, q_a, k)$ with exit state $q_a$ is such that $q_w \in \mathsf{pr}$. Otherwise two processes would be in the same state $q_w$, aiming for the same action state, but at different moments, contradicting Lemma 3.1. Note that we need $|Q_W| + 1$ different identifers, and not $|Q_W|$ for technical

reasons. Finally, the special summary $\texttt{Done}$ is used to indicate when the processes described by a summary reach the exit action state.

We now describe how summaries evolve with the sequence of transitions. Let $S = (\mathsf{pr}, q_a, k)$ and $S' = (\mathsf{pr}', q_a, k)$ be two summaries (with the same exit state and identifier) and $t = (q, !!b, q') \in T$ such that there exists a configuration $C'$ verifying $\overset{\bullet}{q} \oplus \overset{\bullet}{\mathsf{pr}} \xrightarrow{1,t} \overset{\bullet}{q'} \oplus C'$. We then write $S \overset{t}{\Rightarrow} S'$ whenever $\mathbf{set}(C') = \mathsf{pr}' \subseteq Q_W$. This represents the evolution of a summary upon reception of message $b$, when the processes all stay in waiting states, and no new process joins the "waiting region" of the given summary. We write $S \overset{t,+q'}{\Longrightarrow} S'$ whenever $q' \in Q_W$ and $\mathbf{set}(C') \cup \{q'\} = \mathsf{pr}' \subseteq Q_W$. In that latter case, the process responsible for the transition $t$ joins the "waiting region" represented by $S$. This typically occurs when the process's next action state is $q_a$, and it reaches $q_a$ simultaneously with the processes described by $S$. Finally, we use $S \overset{t}{\Rightarrow} \texttt{Done}$ whenever $\mathbf{set}(C') = \{q_a\}$. This represents the evolution (and disappearance) of $S$ when all the processes reach $q_a$ (they all reach it simultaneously).

▶ **Example 3.5.** Returning to the protocol $P$ of Figure 1, consider the summary $S_0 = (\{q_1\}, q_3, 1)$. Observe that $S_0 \xrightarrow{(q_{in},!!a,q_{in})} (\{q_2\}, q_3, 1)$ and $S_0 \xrightarrow{(q_{in},!!a,q_{in})} (\{q_5\}, q_3, 1)$. However, by definition, we do not have $S_0 \xrightarrow{(q_{in},!!a,q_{in})} (\{q_5, q_2\}, q_3, 1)$. Indeed, processes summarized in $S_0$ are forced to all go either on $q_2$ or on $q_5$ upon receiving $a$: thanks to Corollary 3.3, we consider only well-formed executions, where all processes summarized in $S_0$ choose the same next state. Additionnaly, we have $(\{q_2\}, q_3, 1) \xrightarrow{(q_{in},!!b,q_7)} \texttt{Done}$ and $(\{q_1\}, q_4, 1) \xrightarrow{(q_{in},!!a,q_{in})} (\{q_5\}, q_4, 1) \xrightarrow{(q_{in},!!b,q_7),+q_7} (\{q_6, q_7\}, q_4, 1) \xrightarrow{(q_{in},!!d,q_1)} \texttt{Done}$. However, note that we do not have $(\{q_2\}, q_4, 1) \xrightarrow{(q_{in},!!b,q_7)} \texttt{Done}$, since the action state $q_3$ reached from $q_2$ upon receiving $b$ is not the exit state $q_4$.

**Definition of the VASS.** We now explain how we use the summaries in the VASS simulating the executions in the network. First we say that a set $\mathcal{S}$ of summaries is *coherent* if for all distinct pairs $(\mathsf{pr}_1, q_1, k_1), (\mathsf{pr}_2, q_5, k_2) \in \mathcal{S}$ such that $q_1 = q_5$, we have $k_1 \neq k_2$ and $\mathsf{pr}_1 \cap \mathsf{pr}_2 = \emptyset$. We denote by $\mathsf{CoSets}$ the set of coherent sets of summaries. For a set of summaries $\mathcal{S}$ we let $\mathbf{lab}(\mathcal{S}) = \{\mathbf{lab}(S) \mid S \in \mathcal{S}\}$. Observe that, when $\mathcal{S}$ is coherent, for a label $(q, k) \in \mathbf{lab}(\mathcal{S})$, there is a unique $S \in \mathcal{S}$ such that $\mathbf{lab}(S) = (q, k)$.

We define the VASS $\mathcal{V}_P$ simulating the protocol $P$ as follows: $\mathcal{V}_P = (\mathsf{Loc}, s_0, \mathsf{X}, \Delta)$, where $\mathsf{Loc} = \mathsf{CoSets} \cup \{s_0\} \cup \{s_f\}$, the set of counters is $\mathsf{X} = \{\mathsf{x}_q \mid q \in Q_A\} \cup \{\mathsf{x}_{(q,k)} \mid q \in Q_A, k \in [1, |Q_W| + 1]\}$, and the set of transitions $\Delta = \Delta_0 \cup (\bigcup_{t \in T} \Delta_t) \cup \Delta_e$, is defined as follows:

- $\Delta_0$ contains exactly the following transitions:
  - $(s_0, \delta_0, s_0)$ where $\delta_0(\mathsf{x}_{q_{in}}) = 1$ and $\delta_0(\mathsf{x}) = 0$ for all other counters;
  - $(s_0, \mathbf{0}, \emptyset)$ where $\mathbf{0}$ is the null vector (note that the empty set is coherent);
  - $(\{S_f\}, \mathbf{0}, s_f) \in \Delta$ where $S_f = (\{q_f\}, q_u, 1)$;
  - $(s_f, \delta_f, s_f) \in \Delta$ where $\delta_f(\mathsf{x}_{(q_u,1)}) = -1$ and $\delta_f(\mathsf{x}) = 0$ for all other counters.
- For $t = (q, !!a, q') \in T$, we have $(\mathcal{S}, \delta, \mathcal{S}') \in \Delta_t$ iff one of the following conditions holds:

a. $q' \in Q_A$ and $\mathcal{S} = \{S_1, \ldots, S_k\}$. Then, for all $1 \leq i \leq k$, there exists $S_i'$ such that $S_i \overset{t}{\Rightarrow} S_i'$ and $\mathcal{S}' = \{S_i' \mid 1 \leq i \leq k\} \setminus \{\texttt{Done}\}$. Moreover, $\delta(\mathsf{x}_q) = -1$, $\delta(\mathsf{x}_{q'}) = 1$ and $\delta(\mathsf{x}) = 0$ for all $\mathsf{x} \in \mathsf{X} \setminus \{\mathsf{x}_q, \mathsf{x}_{q'}\}$. Here we simply update the sets of states populated by processes in waiting states after the transition $t$. Some summaries may have

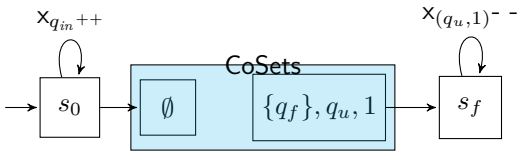


**Figure 2** The structure of the VASS $\mathcal{V}$.

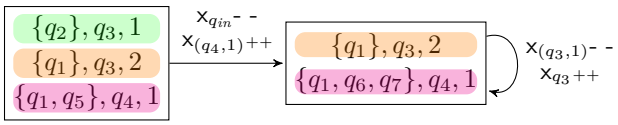


**Figure 3** One successor of location $\{(\{q_2\}, q_3, 1), (\{q_1\}, q_3, 2), (\{q_1, q_5\}, q_4, 1)\}$ (left location) after the broadcast transition $(q_{in}, !!b, q_7)$

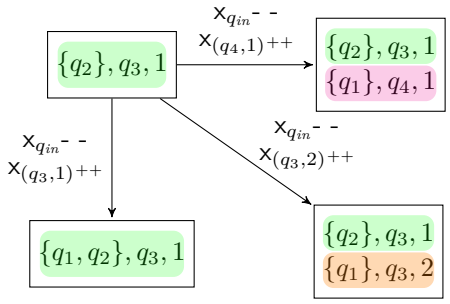


**Figure 4** Three successors of location $\{(\{q_2\}, q_3, 1)\}$ (top left location) after the broadcast transition $(q_{in}, !!d, q_1)$.

disappeared from $\mathcal{S}$ if the processes represented by this summary have reached their exit action state when receiving $a$. For now, we only modify the counters associated to the states $q$ and $q'$. Note that this is well-defined, since we assume in Remark 2.4 that there is no self-loop of the form $(q, !!a, q)$.

b. $q' \in Q_W$, $\mathcal{S} = \{S_1, \dots, S_k\}$, and there exists $1 \leq i \leq k$ and $S_i'$ such that $S_i \xrightarrow{t,+q'} S_i'$. For all $j \neq i$, there exists $S_j'$ such that $S_j \overset{t}{\Rightarrow} S_j'$. Then $\mathcal{S}' = \{S_j' \mid 1 \leq j \leq k\} \setminus \{\texttt{Done}\}$. Moreover, $\delta(\mathsf{x}_q) = -1$, $\delta(\mathsf{x}_{\mathbf{lab}(S_i)}) = 1$ and $\delta(\mathsf{x}) = 0$ for all $\mathsf{x} \in \mathsf{X} \setminus \{\mathsf{x}_q, \mathsf{x}_{\mathbf{lab}(S_i)}\}$. In that case, the process having performed the transition joins an existing summary $S_i$. We have then modified accordingly the counters associated to $q$ and to the appropriate summary.

c. $q' \in Q_W$ and $\mathcal{S} = \{S_1, \dots, S_k\}$. For all $1 \leq j \leq k$, there exists $S_j'$ such that $S_j \overset{t}{\Rightarrow} S_j'$. Then $\mathcal{S}' = (\{S_1', \dots, S_k'\} \setminus \{\texttt{Done}\}) \cup \{(\{q'\}, q_a, k)\}$ for some $q_a \in Q_A$ and $k \in [1, |Q_W| + 1]$ such that $(q_a, k) \notin \mathbf{lab}(\mathcal{S})$. Moreover, $\delta(\mathsf{x}_q) = -1$, $\delta(\mathsf{x}_{(q_a,k)}) = 1$, and $\delta(\mathsf{x}) = 0$ for all $\mathsf{x} \in \mathsf{X} \setminus \{\mathsf{x}_q, \mathsf{x}_{(q_a,k)}\}$. This case happens when the process having sent the message $a$ reaches a waiting state, and its expected next action pair (index and state) does not correspond to any existing summary. In that case, it joins a new summary, the counter associated to the state $q$ is decremented and the counter of this new summary is incremented.

- Finally, $\Delta_e$ allows to empty the counters associated to summaries that have disappeared from the set of locations. It is defined as the set of transitions $(\mathcal{S}, \delta, \mathcal{S}')$ such that: $\mathcal{S} = \mathcal{S}'$ and there exists $(q, k) \notin \mathbf{lab}(\mathcal{S})$ with $\delta(\mathsf{x}_q) = +1$ and $\delta(\mathsf{x}_{(q,k)}) = -1$ and for all other counters, $\delta(\mathsf{x}) = 0$.

Transitions from $\Delta_e$ allow the transfer of counter values from summaries that have disappeared to the counter of their exit action state. This transfer is not immediate, as it is done through iterative decrements and increments of counters. In particular, it is possible for the execution of the VASS to continue without the counter of a disappeared summary being fully transfered. The processes represented by the counter of a disappeared summary behave in the same way as processes in the exit action state that do not take any action. These counters can be emptied later, but always from a location from where no summary with the same label exists. This ensures that there is no ambiguity between processes in a waiting region and those on an action state that have not yet been transferred to the appropriate counter.

▶ **Example 3.6.** Figures 2–4 illustrate the VASS $\mathcal{V}_P$ built for the protocol $P$ from Figure 1. Figure 2 shows its overall structure: any run reaching $(s_f, \mathbf{0})$ starts by incrementing the counter of the initial state and ends by decrementing the counter of the summary $(q_f, q_u, 1)$. Here, $q_u$ is an artificial, unreachable state used to ensure that the counted processes must end in $q_f$ (since they cannot be in $q_u$). Figure 4 illustrates how message receptions and summary creations are handled. The top-left location contains a single summary $(\{q_2\}, q_3, 1)$, representing configurations where all processes on $Q_W$ are in $q_2$, all progressing to $q_3$. After the broadcast $(q_{in}, !!d, q_1)$, three behaviors may follow. In the first case (right), the sender creates a new summary $(\{q_1\}, q_4, 1)$ (pink). In the second (diagonal), it creates another new summary $(\{q_1\}, q_3, 2)$ (orange), indicating a different arrival time at $q_3$. In the third (below), it joins an existing summary, and $q_1$ is added to the print. Well-formed executions restrict the number of summaries with the same exit state, as there are at most $|Q_W|$ distinct moments where processes can reach a given action state (Lemma 3.4). Figure 3 illustrates summary deletion. The transition $(q_{in}, !!b, q_7)$ causes the sender to join the pink summary (bottom one), incrementing its counter. Processes in the green summary $(\{q_2\}, q_3, 1)$ receive the message via $(q_2, ?b, q_3)$, reaching their exit state $q_3$. This summary is deleted in the next location. From there, value of $\mathsf{x}_{(q_3,1)}$ is transferred to the counter $\mathsf{x}_{q_3}$ with the loop transition. If it is not taken enough times, $\mathsf{x}_{(q_3,1)}$ may remain non-zero. This does not affect correctness: $\mathsf{x}_{(q_3,1)}$ will be decremented eventually from a state in which there is no summary labeled with $(q_3, 1)$. Not decrementing soon enough only delay the moment the corresponding processes will move from the action state $q_3$.

▶ Remark 3.7. In the sequel of this paper, the size of $\mathcal{V}_P$ will be of interest to us. Hence, observe that $|\mathsf{Loc}| = |\mathsf{CoSets}|+3 \leq 2^{|Q_A|\times(|Q_W|+1)\times 2^{|Q_W|}}+3$ (as one summary is composed of a state in $Q_A$, one label in $[1, |Q_W|+1]$ and one set of waiting states), and $|\mathsf{X}| = |Q_A|+|Q_A|\times(|Q_W|+1)$.

**Soundness of the construction.** We show now that if there exists a run in the VASS $\mathcal{V}_P$ from $(s_0, \mathbf{0})$ to $(s_f, \mathbf{0})$, then in the network built from $P$, there exist $C \in \mathcal{I}$ and $C' \in \mathcal{C}$ such that $C \to^* C'$ and $C'(e) = q_f$ for all $e \in [1, ||C'||]$.

We say that a configuration $(\ell, v)$ of $\mathcal{V}_P$ is an *S-configuration* if $\ell = \mathcal{S}$ for some $\mathcal{S} \in \mathsf{CoSets}$. The *implementation* of an S-configuration $\lambda = (\mathcal{S}, v)$ is the set of network configurations $[\![\lambda]\!] \subseteq \mathcal{C}$ defined as follows: $C \in [\![\lambda]\!]$ if and only if there exists a function $f : [1, ||C||] \to \mathsf{X}$ such that $|f^{-1}(\mathsf{x})| = v(\mathsf{x})$ for all $\mathsf{x} \in \mathsf{X}$, and

- **CondImpl1** for all $\mathsf{x}_q \in \mathsf{X}$ with $q \in Q_A$, we have $C(e) = q$ for all $e \in f^{-1}(\mathsf{x}_q)$,;
- **CondImpl2** for all $\mathsf{x}_{(q,k)} \in \mathsf{X}$, if there exists $(\mathsf{pr}, q, k) \in \mathcal{S}$ for some $\mathsf{pr} \subseteq Q_W$, then $C(e) \in \mathsf{pr} \cup \{q\}$ for all $e \in f^{-1}(\mathsf{x}_{q,k})$;
- **CondImpl3** for all $\mathsf{x}_{(q,k)} \in \mathsf{X}$, if $(q, k) \notin \mathbf{lab}(\mathcal{S})$, then $C(e) = q$ for all $e \in f^{-1}(\mathsf{x}_{q,k})$.

In an implementation of $\lambda$, the processes populate states according to the values of the counters. All processes associated with a counter $\mathsf{x}_q$ of an action state will populate this exact state (**CondImpl1**). However, processes associated with a counter $\mathsf{x}_{(q,k)}$ of a summary do not necessarily populate waiting states. This occurs when the label $(q, k)$ does not appear in $\mathcal{S}$ while the associated counter remains strictly positive. Such a situation arises when a summary has been previously deleted, but its counter has not been emptied. Since all associated processes have already reached the exit action state, the processes associated to the corresponding counter have to populate this active state (**CondImpl3**). If the summary with label $(q, k)$ is active (i.e. its label appears in $\mathcal{S}$), the processes associated with $\mathsf{x}_{(q,k)}$ will be in the corresponding waiting region, or in the exit action state (**CondImpl2**). This may seem contradictory, as all processes are supposed to reach their exit state simultaneously.

However, the processes already on the exit action state are "old" processes, that remained after a previous deletion of this summary (cf. **CondImpl3**). These processes will be allowed to send messages only when they are identified with the counter of the active state, meaning when the corresponding transition in $\Delta_e$ will be taken. The next lemma allows to build an execution of the protocol $P$ from an execution of the VASS $\mathcal{V}_P$.

▶ **Lemma 3.8.** *Let $\lambda, \lambda'$ be two S-configurations of $\mathcal{V}_P$ and $C \in [\![\lambda]\!]$. If $\lambda \overset{\delta}{\rightsquigarrow\!\triangleright} \lambda'$ in $\mathcal{V}_P$ then there exists $C' \in [\![\lambda']\!]$ such that $C \to^* C'$ for $P$.*

**Sketch of Proof.** Assume $\lambda = (\mathcal{S}, v)$ and $\lambda' = (\mathcal{S}', v')$. Then either $(\mathcal{S}, \delta, \mathcal{S}') \in \Delta_e$ or $(\mathcal{S}, \delta, \mathcal{S}') \in \Delta_t$ for some $t \in T$. In the first case, we show that if $C \in [\![\lambda]\!]$, then $C \in [\![\lambda']\!]$. Otherwise, let $t = (q, !!a, q')$, there is a process $e$ such that $C(e) = q$. The transition $\delta$ in the VASS makes all the summaries and counters evolve according to the reception of the message $a$. According to the different cases a., b. or c., one can build a configuration $C' \in [\![\lambda']\!]$ such that $C \to C'$. ◀

▶ **Lemma 3.9.** *If $(s_0, \mathbf{0}) \rightsquigarrow\!\triangleright^* (s_f, \mathbf{0})$ in $\mathcal{V}_P$, then there exist $C \in \mathcal{I}$ and $C' \in \mathcal{C}$ such that $C \to^* C'$ and $C'(e) = q_f$ for all $e \in [1, ||C'||]$.*

**Proof.** From the definition of $\mathcal{V}_P$, any run from $(s_0, \mathbf{0})$ to $(s_f, \mathbf{0})$ is of the form $(s_0, \mathbf{0}) \rightsquigarrow\!\triangleright^* (\emptyset, v_{init}) \rightsquigarrow\!\triangleright^* (\{S_f\}, v_f) \rightsquigarrow\!\triangleright^* (s_f, \mathbf{0})$ where $v_{init}(\mathsf{x}_{q_{in}}) = n$ for some $n \in \mathbb{N}$ and $v_{init}(\mathsf{x}) = 0$ for all $\mathsf{x} \in \mathsf{X} \setminus \{\mathsf{x}_{q_{in}}\}$, whereas $v_f(\mathsf{x}_{(q_u,1)}) = v_{init}(\mathsf{x}_{q_{in}}) = n$ and $v_f(\mathsf{x}) = 0$ for all $\mathsf{x} \in \mathsf{X} \setminus \{\mathsf{x}_{(q_u,1)}\}$. Define $C$ as the initial configuration in $\mathcal{I}$ with $n$ processes (i.e. $||C|| = n$). Trivially, $C \in [\![(\emptyset, v_{init})]\!]$ and with Lemma 3.8 and a simple induction, we get that there exists $C' \in [\![(\{S_f\}, v_f)]\!]$ such that $C \to^* C'$. Observe that by definition of $v_f$, and since $C' \in [\![(\{S_f\}, v_f)]\!]$, there exists $f : [1, n] \to \mathsf{X}$ such that $f^{-1}(\mathsf{x}_{(q_u,1)}) = [1, n]$. Using **CondImpl2**, we have $C'(e) \in \{q_f, q_u\}$ for all $e \in [1, n]$ with $n = ||C'||$. Since $q_u$ is unreachable by construction, we deduce that $C'(e) = q_f$ for all $e \in [1, ||C'||]$. ◀

**Completeness of the construction.** Let us first introduce some new definitions. Given a well-formed execution $\rho \in \Lambda \cup \Lambda_\omega$, two indices $0 \leq i < j < |\rho|$ and an action state $q_a \in Q_A$, we define $E^{\rho,i}_{q_a,j}$, the set of processes in waiting states in $\rho_i$, and whose next action state is $q_a$, reached at $\rho_j$. Formally, $E^{\rho,i}_{q_a,j} = \{e \in [1, \#\mathbf{proc}(\rho)] \mid \rho_i(e) \in Q_W \text{ and } \mathbf{na}(\rho, i, e) = (q_a, j)\}$. Furthermore, an S-configuration $\lambda = (\mathcal{S}, v)$ is a *representative* of the configuration $\rho_i$ in $\rho$ iff the following conditions are respected:

**CondRepr1** for all $q \in Q_A$, $v(\mathsf{x}_q) = |\rho_i^{-1}(q)|$, and

for all $q_a \in Q_A$, there is an injective function $r_{q_a} : \mathbf{na\text{-}index\text{-}set}(\rho, i, q_a) \to [1, |Q_W| + 1]$ s.t.:

**CondRepr2** $\mathcal{S} = \{(\rho_i(E^{\rho,i}_{q_a,j}), q_a, r_{q_a}(j)) \mid q_a \in Q_A, j \in \mathbf{na\text{-}index\text{-}set}(\rho, i, q_a)\}$

**CondRepr3** for all $q_a \in Q_A$, we have $v(\mathsf{x}_{(q_a, r_{q_a}(j))}) = |E^{\rho,i}_{q_a,j}|$ for all $j \in \mathbf{na\text{-}index\text{-}set}(\rho, i, q_a)$ and $v(\mathsf{x}_{(q_a,k)}) = 0$ for all $k \notin r_{q_a}(\mathbf{na\text{-}index\text{-}set}(\rho, i, q_a))$.

In a representative of the configuration $\rho_i$, the counters faithfully count the number of processes on active states (**CondRepr1**), and on waiting regions. The set $E^{\rho,i}_{q_a,j}$ gathers exactly the set of processes that populate a same waiting region in a representative of $\rho_i$ in $\rho$: they all reach the same next action state $q_a$ at the same instant $j$. The set $\mathbf{na\text{-}index\text{-}set}(\rho, i, q_a)$ being bounded by $|Q_W|$ (cf. Lemma 3.4), we can associate with each of these indices a unique identifier, given by the injective function $r_{q_a}$. We use a spare identifier for technical reasons, to handle more easily situations when $|Q_W|$ summaries exist in the location of the VASS, and a new summary is created while one of the previous

summaries is deleted at the next step. Then, **CondRepr2** and **CondRepr3** require that the counters corresponding to the summaries and the summaries themselves reflect faithfully the situation in the configuration $\rho_i$ with respect to $\rho$. Observe that such a representative is tightly linked to the simulated execution, since we need to know the future behavior of the different processes to determine the different summaries. Finally, if we let $\mathbf{repr}(\rho, i)$ be the set of all $S$-configurations that are representatives of $\rho_i$ in $\rho$, we establish the following result before proving the main lemma of this subsection (Lemma 3.11).

▶ **Lemma 3.10.** *Let $\rho$ be a* well-formed *execution, and $0 \leq i < |\rho|$. For all $\lambda_i \in \mathbf{repr}(\rho, i)$, there exists $\lambda_{i+1} \in \mathbf{repr}(\rho, i+1)$ such that $\lambda_i \rightsquigarrow_{\triangleright}^* \lambda_{i+1}$ in $\mathcal{V}_P$.*

**Sketch of Proof.** From an S-configuration $\lambda_i = (\mathcal{S}_i, v_i)$ in $\mathbf{repr}(\rho, i)$, we can compute the effect on $\mathcal{S}_i$ of the transition $t = (q, !!a, q')$ taken in $\rho$ to go from $\rho_i$ to $\rho_{i+1}$, and find a matching transition $(\mathcal{S}_i, \delta, \mathcal{S}_{i+1}) \in \Delta_t$ (according to whether $q'$ is a waiting state or not, and in the latter case, whether it joins an existing summary or a new one, depending of the execution $\rho$). We then obtain a new configuration $\lambda' = (\mathcal{S}_{i+1}, v')$ such that $\lambda_i \overset{\delta}{\rightsquigarrow}_{\triangleright} \lambda'$. Note that $\lambda'$ is not necessarily in $\mathbf{repr}(\rho, i+1)$: if some summaries have been deleted between $\lambda_i$ and $\lambda'$, the transition $(\mathcal{S}_i, \delta, \mathcal{S}_{i+1})$ in $\mathcal{V}_P$ has not updated the corresponding counters. Hence we need to apply transitions from $\Delta_e$ to empty counters corresponding to deleted summaries and accordingly increase corresponding counters on matching action states to obtain $\lambda_{i+1} = (\mathcal{S}_{i+1}, v_{i+1})$ in $\mathbf{repr}(\rho, i+1)$ . ◀

▶ **Lemma 3.11.** *If there exist $C \in \mathcal{I}$, and $C' \in \mathcal{C}$ such that $C \to^* C'$ in $P$ and $C'(e) = q_f$ for all $e \in [1, ||C'||]$, then $(s_0, \mathbf{0}) \rightsquigarrow_{\triangleright}^* (s_f, \mathbf{0})$ in $\mathcal{V}_P$.*

**Proof.** Thanks to Corollary 3.3, we know there exists a well-formed execution $\rho$ from $C$ to $C'$. Let $K = ||C||$ and $n = |\rho| - 1$. First, consider the sequence $(s_0, \mathbf{0}) \rightsquigarrow_{\triangleright}^* (s_0, v_0) \rightsquigarrow_{\triangleright} (\emptyset, v_0)$ where $v_0(\mathsf{x}_{q_{in}}) = K$ and for all other counters $\mathsf{x}$, $v_0(\mathsf{x}) = 0$. Observe that, since $q_{in} \in Q_A$, $(\emptyset, v_0) \in \mathbf{repr}(\rho, 0)$. By applying inductively Lemma 3.10, we get that $(\emptyset, v_0) \rightsquigarrow_{\triangleright}^* (\mathcal{S}_1, v_1) \rightsquigarrow_{\triangleright}^* \cdots \rightsquigarrow_{\triangleright}^* (\mathcal{S}_n, v_n)$ with $(\mathcal{S}_n, v_n) \in \mathbf{repr}(\rho, n)$. By definition, every time that $\mathbf{na\text{-}state}(\rho, i, e) = q_u$ for some process $e$, then $\mathbf{na\text{-}index}(\rho, i, e) = n + 1$, hence there will always be at most one summary with exit state $q_u$. So in the execution we build, every time that some summary $(\mathsf{pr}, q_u, k) \in \mathcal{S}_i$, we chose $k = 1$. We then have $v_n(\mathsf{x}_{q_u,1}) = K$ and $v_n(\mathsf{x}) = 0$ for all other $\mathsf{x} \in \mathsf{X} \setminus \{\mathsf{x}_{q_u,1}\}$. Hence, by construction of $\mathcal{V}_P$, we have $(\mathcal{S}_n, v_n) \rightsquigarrow_{\triangleright} (s_f, v_n) \rightsquigarrow_{\triangleright}^* (s_f, \mathbf{0})$. ◀

Using Lemma 3.10 and Lemma 3.11 and the fact that the reachability problem for VASS is decidable (see Theorem 2.5) we get the main theorem of this section.

▶ **Theorem 3.12.** *Synchro restricted to Wait-Only protocols is decidable.*

## 3.3 Lower Bound for Synchro in Wait-Only Protocols

In this subsection we reduce the reachability problem for VASS to Synchro. We fix a VASS $\mathcal{V} = (\mathsf{Loc}, \ell_0, \mathsf{X}, \Delta)$ and a final location $\ell_f \in \mathsf{Loc}$. W.l.o.g., we suppose that any transition in $\Delta$ either increments or decrements of 1 exactly one counter. Hence, for a transition $(\ell, \delta, \ell') \in \mathsf{Loc}$, we might describe $\delta$ by giving only $\delta(\mathsf{x})$ for the only $\mathsf{x} \in \mathsf{X}$ such that $\delta(\mathsf{x}) \neq 0$.

We explain how to build a protocol $P_{\mathcal{V}}$ that simulates $\mathcal{V}$: it is depicted in Figure 5 *in which we don't consider the dashed edge*. The states $\mathsf{zero}$ and $\{\mathsf{unit}_{\mathsf{x}} \mid \mathsf{x} \in \mathsf{X}\}$ represent the evolution of the different counters of the VASS while the blue box $P'_{\mathcal{V}}$ describes the evolution of $\mathcal{V}$ with respect to its locations. Formally, $P'_{\mathcal{V}}$ consists of a set of states $Q_{\mathcal{V}} = \mathsf{Loc} \cup \{☹\}$

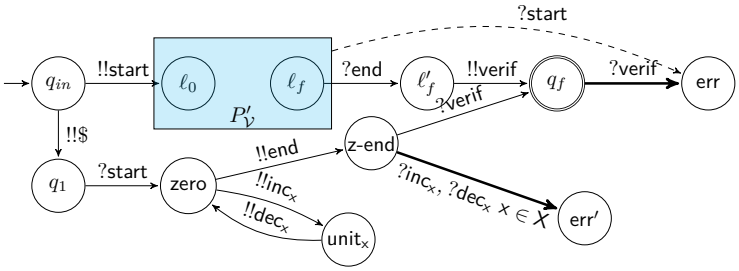


**Figure 5** Protocol $P$ associated to a VASS $\mathcal{V}$. The dashed edge $(\ell_f, ?\mathsf{start}, \mathsf{err})$ will only be considered in Section 4.2.


**Figure 6** Part of $P'_{\mathcal{V}}$ that simulates respectively the transition $(\ell, \delta, \ell') \in \Delta$ with $\delta(x) = 1$ at the left, and $(\ell'', \delta, \ell''') \in \Delta$ with $\delta(x) = -1$ at the right.

and a set of transitions $T_{\mathcal{V}} = \{(\ell, ?\mathsf{inc}_x, \ell') \mid (\ell, \delta, \ell') \in \Delta \text{ and } \delta(x) = 1\} \cup \{(\ell, ?\mathsf{dec}_x, \ell') \mid (\ell, \delta, \ell') \in \Delta \text{ and } \delta(x) = -1\} \cup \{(\ell, ?m, ☹) \mid m \in \{\mathsf{inc}_x, \mathsf{dec}_x \mid x \in X\}\}$. Some transitions of $T_{\mathcal{V}}$ are depicted in Figure 6. We then obtain the following theorem.

▶ **Theorem 3.13.** *Synchro with Wait-Only protocols is Ackermann-complete.*

**Sketch of Proof.** The upper bound follows from Theorem 3.12 and Theorem 2.5. For the lower bound, we give the ideas that prove that the reduction described above is correct. In particular, there exist $C_0 \in \mathcal{I}$ and $C_f \in \mathcal{C}$ such that $C_0 \to^* C_f$ in $P_{\mathcal{V}}$ and $C_f(e) = q_f$ for all $e \in [1, ||C_f||]$ *if and only if* $(\ell_0, \mathbf{0}) \rightsquigarrow^* (\ell_f, \mathbf{0})$ in $\mathcal{V}$.

Assume that $(\ell_0, \mathbf{0}) \rightsquigarrow^* (\ell_f, \mathbf{0})$ in $\mathcal{V}$, and let $K \in \mathbb{N}$ be the maximum value of $\sum_{x \in X} v(x)$ reached during the run. We choose $C_0$ such that $||C_0|| = K + 1$. The execution proceeds as follows: all processes except one (the leader) broadcast a dummy message \$. The leader then broadcasts start and reaches $\ell_0$, while the others move to zero. These simulate the counter values, and the leader simulates the VASS by moving through the states of $P'_{\mathcal{V}}$. The value of counter x is represented by the number of processes in $\mathsf{unit}_x$. To simulate an increment of x from $(\ell_i, v_i)$ to $(\ell_{i+1}, v_{i+1})$, a process in zero sends $\mathsf{inc}_x$, which the leader receives to transition to $\ell_{i+1}$. If the current configuration correctly represents $(\ell_i, v_i)$, the new one faithfully represents $(\ell_{i+1}, v_{i+1})$. At the end of the simulation, the leader reaches $\ell_f$, and the other processes remain in zero. They then broadcast end, and the leader (now in $\ell'_f$) broadcasts verif, gathering all processes in $q_f$.

To prove the other direction, we must show that the processes cannot cheat in simulating the VASS. First, note that reaching $q_f$ requires exactly one broadcast of verif, a second would send the original sender to err. This ensures that only one process (the leader) simulates the VASS by moving through $P_{\mathcal{V}}$. Second, counter updates must follow the VASS transitions: any unauthorized update would cause the leader to receive an unexpected message and transition to the losing state ☹. Finally, when the leader reaches $\ell_f$, all other processes must be in zero. If not, problems occur during the final steps: one process (the helper) broadcasts end, which sends the leader to $\ell'_f$. Then, all remaining processes must reach z-end before the unique verif broadcast. If a process in $\mathsf{unit}_x$ moves to z-end, it must broadcast $\mathsf{dec}_x$, which is

received by other processes in z-end, sending them, including the helper, to err′. As a result, the helper misses verif and cannot reach $q_f$, preventing a successful execution. Thus, the simulation must end with the leader in $\ell_f$ and all others in zero. ◀

# 4 Synchro is ExpSpace-complete when the Target State is an Action State

**Upper bound.** We show that Synchro when $q_f \in Q_A$ can be reduced to the mutual reachability problem on VASS, defined as follows: given a VASS $\mathcal{V} = (\mathsf{Loc}, \ell_0, \mathsf{X}, \Delta)$ and a location $\ell_f$, do there exist a run from $(\ell_0, \mathbf{0})$ to $(\ell_f, \mathbf{0})$ and one from $(\ell_f, \mathbf{0})$ to $(\ell_0, \mathbf{0})$. This problem is shown to be in ExpSpace [16]. More precisely, Leroux establishes the ExpSpace-membership and provides a bound on the lengths of the runs, for the mutual reachability problem in VAS (Vector Addition Systems). However, by applying the VASS-to-VAS transformation presented in [14], we get the following result.

▶ **Theorem 4.1** ([16] Corollary 10.6, Transformation of [14]). *Given a VASS $\mathcal{V} = (\mathsf{Loc}, \ell_0, \mathsf{X}, \Delta)$, if there are two runs $(\ell_0, \mathbf{0}) \rightsquigarrow_{\triangleright}^* (\ell_f, \mathbf{0})$ and $(\ell_f, \mathbf{0}) \rightsquigarrow_{\triangleright}^* (\ell_0, \mathbf{0})$, then there are two runs $(\ell_0, \mathbf{0}) \rightsquigarrow_{\triangleright}^* (\ell_f, \mathbf{0})$ and $(\ell_f, \mathbf{0}) \rightsquigarrow_{\triangleright}^* (\ell_0, \mathbf{0})$ whose lengths are bounded by $17(|\mathsf{X}| + 3)^2 x^{15(|\mathsf{X}|+3)^{|\mathsf{X}|+5}}$ where $x = (1 + 2(|\mathsf{Loc}| + 1)^2)^2$.*

Given a Wait-Only protocol $P = (Q, \Sigma, q_{in}, T)$ and a target state $q_f \in Q_A$, we explain how to transform it into a VASS where mutual reachability is equivalent to Synchro. We build the VASS $\mathcal{V}_P = (\mathsf{Loc}, \mathsf{X}, s_0, \Delta)$ as described in Section 3.2, with some modifications. The first two modifications simply encode the fact that the target state is not a waiting sate anymore, while the third one ensures the mutual reachability.

- We add a state $s'_f$ and the transition $(\{S_f\}, \mathbf{0}, s_f)$ is replaced by two transitions: $(\emptyset, \delta, s'_f)$ and $(s'_f, \delta', s_f)$. Here, $\delta(\mathsf{x}_{q_f}) = -1$, $\delta'(\mathsf{x}_{q_f}) = 1$, and for all other counters $\mathsf{x}$, $\delta(\mathsf{x}) = \delta'(\mathsf{x}) = 0$. It prevents the reachability of $(s_f, \mathbf{0})$ to be trivial with the run $(s_0, \mathbf{0}) \rightsquigarrow_{\triangleright} (\emptyset, \mathbf{0}) \rightsquigarrow_{\triangleright} (s_f, \mathbf{0})$.
- The transition $(s_f, \delta_f, s_f)$ is replaced by $(s_f, \delta'_f, s_f)$, where $\delta'_f(\mathsf{x}_{q_f}) = -1$ and $\delta'_f(\mathsf{x}) = 0$ for all other counters $\mathsf{x}$.
- We add a transition $(s_f, \mathbf{0}, s_0)$ to allow the resetting of $\mathcal{V}_P$ to $s_0$ after reaching $s_f$.

▶ **Lemma 4.2.** *There exists $C \in \mathcal{I}$ and $C' \in \mathcal{C}$ such that $C \to^* C'$ and, for all $e \in [1, ||C'||]$, $C'(e) = q_f$ if and only if there are two runs $(s_0, \mathbf{0}) \rightsquigarrow_{\triangleright}^* (s_f, \mathbf{0})$ and $(s_f, \mathbf{0}) \rightsquigarrow_{\triangleright}^* (s_0, \mathbf{0})$ in $\mathcal{V}_P$.*

**Sketch of Proof.** Assume that there exist $C \in \mathcal{I}$ and $C' \in \mathcal{C}$ such that $C \to^* C'$ and, for all $e \in [1, ||C'||]$, $C'(e) = q_f$. Since the main body of $\mathcal{V}_P$ is the same as in Section 3.2 we can, like in Lemma 3.11, build a run of $\mathcal{V}_P$ from $(s_0, \mathbf{0})$ to $(s_f, \mathbf{0})$. By construction of $\mathcal{V}_P$, $(s_f, \mathbf{0}) \rightsquigarrow_{\triangleright} (s_0, \mathbf{0})$.Assume now that $(s_0, \mathbf{0}) \rightsquigarrow_{\triangleright}^* (s_f, \mathbf{0})$ (and $(s_f, \mathbf{0}) \rightsquigarrow_{\triangleright} (s_0, \mathbf{0})$). Observe that it might be the case that the run looks like $(s_0, \mathbf{0}) \rightsquigarrow_{\triangleright}^* (s_f, v_1) \rightsquigarrow_{\triangleright} (s_0, v_1) \rightsquigarrow_{\triangleright}^* (s_f, v_2) \dots (s_0, v_k) \rightsquigarrow_{\triangleright}^* (s_f, \mathbf{0})$. By construction of the VASS, we know that there is an execution $(s_0, \mathbf{0}) \rightsquigarrow_{\triangleright}^* (\emptyset, v'_0) \rightsquigarrow_{\triangleright}^* (\emptyset, v''_0) \rightsquigarrow_{\triangleright} (s'_f, v'''_0) \rightsquigarrow_{\triangleright} (s_f, v''_0) \rightsquigarrow_{\triangleright}^* (s_f, v_1)$. Here $v'_0$ is the valuation obtained after having increased the counter $\mathsf{x}_{q_{in}}$, $v''_0$ is the valuation obtained just before going to $s'_f$, and $v_1$ is obtained after having decremented counter $\mathsf{x}_{q_f}$. Adapting the proof of Lemma 3.9 to this case, we deduce the existence of an execution of the protocol from an initial configuration $C_0$ to a configuration $C'_0$ such that $|{C'_0}^{-1}(q)| = 0$ for all $q \in Q_W$, and $|{C'_0}^{-1}(q)| = v''_0(\mathsf{x}_q) + \sum_{i=1,\dots,|Q_W|+1} v''_0(\mathsf{x}_{(q,i)})$ for all $q \in Q_A$. From the portion of run of the VASS $(s_0, v_1) \rightsquigarrow_{\triangleright}^* (\emptyset, v'_1) \rightsquigarrow_{\triangleright}^* (\emptyset, v''_1) \rightsquigarrow_{\triangleright} (s'_f, v'''_1) \rightsquigarrow_{\triangleright} (s_f, v''_1) \rightsquigarrow_{\triangleright}^* (s_f, v_2)$, we

similarly get another sequence of configurations of the protocol from a configuration $C_1$ (not necessarily initial because $v_1$ might not be equal to 0 for some counters other than $\mathsf{x}_{q_{in}}$) to a configuration $C_1'$ such that $C_1 \to^* C_1'$ and $|{C_1'}^{-1}(q)| = 0$ for all $q \in Q_W$. Observe however that these two sequences can be merged into a single one, of size $v_0'(\mathsf{x}_{q_{in}}) + (v_1'(\mathsf{x}_{q_{in}}) - v_1(\mathsf{x}_{q_{in}}))$ (the second part corresponds to processes added in $q_{in}$ with transition $(s_0, \delta_{in}, s_0)$, and $\delta_{in}(\mathsf{x}_{q_{in}}) = 1$). The execution then first behaves like the execution from $C_0$ to $C_0'$, potentially leaving some processes in the initial state (in particular processes from $v_1'(\mathsf{x}_{q_{in}}) - v_1(\mathsf{x}_{q_{in}})$). Once this first part is over, all the processes are either in the initial state, or in an action state (because ${C_0'}^{-1}(q) = 0$ for all $q \in Q_W$). Then, one can simulate the second sequence, (processes already in $q_f$ from the first execution won't be affected because they are in an action state, so they won't receive any message. This would not be true if $q_f \in Q_W$). Since the execution of $\mathcal{V}_P$ eventually reaches $(s_f, \mathbf{0})$, it means that it reaches $(s_f, v_{k+1})$ where $v_{k+1}(\mathsf{x}) = 0$ for all $\mathsf{x} \neq \mathsf{x}_{q_f}$. Then, by iterating the construction described above, one obtains an execution of $P$ from an initial configuration to a configuration $C_f$ such that $C_f(e) = q_f$ for all $e \in [1, ||C_f||]$. ◀

As runs of doubly exponential length can be guessed in exponential space in the size of the VASS and ExpSpace = NExpspace, we get the following theorem using Remark 3.7 and Theorem 4.1. Observe that even if the number of locations is doubly exponential in the size of the protocol, the bound of Theorem 4.1 is polynomial in $|\mathsf{Loc}|$ and doubly exponential in $|\mathsf{X}|$, hence lengths of the runs to guess remain doubly exponential in the size of the protocol.

▶ **Theorem 4.3.** *Synchro for Wait-Only protocols is in ExpSpace when $q_f \in Q_A$.*

**Lower bound.** ExpSpace-hardness follows from a reduction of the coverability problem in VASS, which is ExpSpace-hard [19], and stated as follows: given a VASS $\mathcal{V} = (\mathsf{Loc}, \ell_0, \mathsf{X}, \Delta)$ of dimension $d \in \mathbb{N}$, and a location $\ell_f \in \mathsf{Loc}$, decide whether there is an execution from $(\ell_0, \mathbf{0})$ to $(\ell_f, v)$ for some $v \in \mathbb{N}^d$.

The reduction uses the protocol depicted in Figure 5, this time including the dashed edge but excluding the thick edges: $q_f$ is now an action state. Then $\ell_f$ is coverable iff there exists an execution $C_0 \to^* C_f$ in which all processes reach $q_f$. The execution of the VASS can be simulated in the protocol like in Section 3.3. The processes that may still remain in the states $\mathsf{unit}_\mathsf{x}$ can reach $\mathsf{z\text{-}end}$ even when $\ell_f'$ is populated (recall that all the thick edges have been removed). Once this is done, all processes can gather in $q_f$. Conversely, if there exists an execution in which all processes reach $q_f$, it means that $\ell_f$ is coverable. Let $e_0$ be the first process to broadcast $\mathsf{start}$, then an execution of the VASS covering $\ell_f$ can be built based on the sequence of states visited by $e_0$ between $\ell_0$ and $\ell_f$. This relies on three keys observations: (1) The process $e_0$ must visit $\ell_f$, otherwise it cannot reach $q_f$. (2) When $e_0$ is between $\ell_0$ and $\ell_f$, $e_0$ is the only process in this region. If another process attempts to join by broadcasting $\mathsf{start}$, $e_0$ will receive the message and go to $\mathsf{err}$, preventing it from reaching $q_f$. (3) When $e_0$ reaches $\ell_0$ for the first time, no other process is in the states $\{\mathsf{unit}_\mathsf{x} \mid \mathsf{x} \in \mathsf{X}\}$, since $\mathsf{start}$ is sent for the first time at this point. Hence, with Theorem 4.3, it establishes the following result.

▶ **Theorem 4.4.** *Synchro for Wait-Only protocols is ExpSpace-complete when $q_f \in Q_A$.*

## 5 Solving the Repeated Coverability problem for Wait-Only Protocols

**Upper Bound.** We show that RepCover on Wait-Only protocols is in ExpSpace, via the repeated coverability problem in VASS: given a VASS $\mathcal{V} = (\mathsf{Loc}, \ell_{init}, \mathsf{X}, \Delta)$ and a location

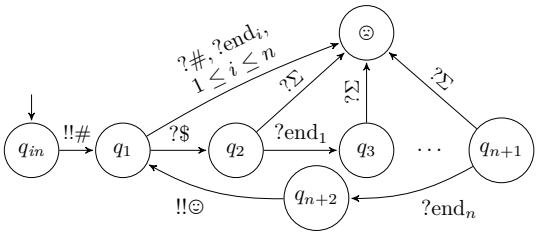


**Figure 7** The part of $P$ with transition $t_f = (q_{n+2}, !!☺, q_1)$. We write ?Σ for $?m, \forall m \in \Sigma$.

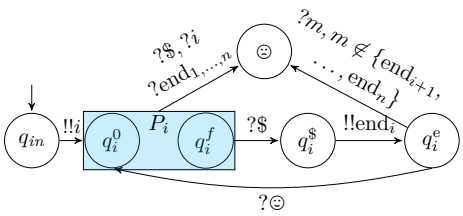


**Figure 8** The part of $P$ simulating automaton $i$.

$\ell_f \in \mathsf{Loc}$, is there an infinite execution $(\ell_{init}, \mathbf{0}) \rightsquigarrow (\ell_1, v_1) \rightsquigarrow \dots$ such that, for all $i \in \mathbb{N}$, there exists $j > i$ such that $\ell_j = \ell_f$? We rely on the following theorem.

▶ **Theorem 5.1** (Theorem 3.1 of [13]). *For a VASS $\mathcal{V} = (\mathsf{Loc}, \ell_{init}, \mathsf{X}, \Delta)$, the repeated coverability problem is solvable in $O(\log(l) + \log(|\mathsf{Loc}|))2^{c|\mathsf{X}|\log(|\mathsf{X}|)}$ nondeterministic space for some constant $c$ independent of $\mathcal{V}$, and where the absolute values of components of vectors in $\Delta$ are smaller than $l$.*

Let $P = (Q, \Sigma, q_{in}, T)$ be a Wait-Only protocol and $t_f = (q, !!a, q')$ the transition that has to occur infinitely often. We build a VASS $\mathcal{V}_P$ based on the construction presented in Section 3.2. This time, we add a new set of states $\mathsf{CoSets}_{☺} = \{(\mathcal{S}, ☺) \mid \mathcal{S} \in \mathsf{CoSets}\}$. and the set of transitions $\{(\mathcal{S}, \delta, (\mathcal{S}', ☺)) \mid (\mathcal{S}, \delta, \mathcal{S}') \in \Delta_{t_f}\} \cup \{(\mathcal{S}, ☺), \mathbf{0}, \mathcal{S}) \mid \mathcal{S} \in \mathsf{CoSets}\}$. Hence, when a process takes the transition $t_f$, it is highlighted in $\mathcal{V}_P$ by going in a location tagged with ☺, before continuing the execution. Taking the protocol of Figure 1, with $t_f = (q_{in}, !!d, q_1)$, the previous transition from the location of the VASS $\{(\{q_3\}, q_4, 1)\}$ to the location $\{(\{q_3\}, q_4, 1), (\{q_1\}, q_6, 1)\}$ (see Figure 4) is now transformed into two transitions in the VASS we build here: one from $(\{(\{q_3\}, q_4, 1)\})$ to the location $(\{(\{q_3\}, q_4, 1), (\{q_1\}, q_6, 1)\}, ☺)$ and one from $(\{(\{q_3\}, q_4, 1), (\{q_1\}, q_6, 1)\}, ☺)$ to $\{(\{q_3\}, q_4, 1), (\{q_1\}, q_6, 1)\}$. There is an execution of $P$ with a process that takes $t_f$ infinitely often iff there is an execution of $\mathcal{V}_P$ where one state in $\mathsf{CoSets}_{☺}$ is visited infinitely often. The procedure to decide RepCover is then: guess a location $\ell_f$ in $\mathsf{CoSets}_{☺}$ and solve the repeated coverability problem for $\mathcal{V}_P$ and $\ell_f$. From Remark 3.7 and Theorem 5.1, this can be done in $O(\log(2) + (|Q| + 1)^2 \times 2^{|Q|} + 2)2^{c(|Q|+1)^3 \log((|Q|+1)^3)}$ nondeterministic space. The overall procedure is then in NExpSpace = ExpSpace. Hence, the following theorem.

▶ **Theorem 5.2.** *RepCover is in ExpSpace.*

▶ Remark 5.3. One could define RepCover with a reception transition that has to occur infinitely often. The VASS we described hereabove can be adapted by enlarging each location with the current state of the witness process. This can also be done in ExpSpace.

**Lower bound.** We reduce the intersection non-emptiness problem for deterministic finite automata, which is known to be PSpace-complete [15], to RepCover. Let $\mathcal{A}_1, \dots, \mathcal{A}_n$ be deterministic, complete finite automata, with $\mathcal{A}_i = (A, Q_i, q_i^0, q_i^f, \Delta_i)$ for $1 \le i \le n$. The reduction assumes a single accepting state per automaton, which does not affect complexity. We denote by $A^*$ the set of words over the alphabet $A$, and extend each transition function $\Delta_i$ to $A^*$ by defining: $\Delta_i^*(q, \varepsilon) = q$, and $\Delta_i^*(q, wa) = \Delta_i(\Delta_i^*(q, w), a)$ for all $w \in A^*$ and $a \in A$.

We define a protocol $P = (Q, \Sigma, q_{in}, T)$ composed of several components. For each automaton $\mathcal{A}_i$, $P$ includes a component $P_i = (Q^i, T^i)$ (see Figure 8), where $Q^i = Q_i$ and

$T^i = \{(q_1^i, ?a, q_2^i) \mid (q_1^i, a, q_2^i) \in \Delta_i\}$. The main control component (see Figure 7) includes the transition $t_f = (q_{n+2}, !!\text{☺}, q_1)$, which must occur infinitely often. A process taking $t_f$ infinitely often also receives infinitely many sequences of messages $\$ \cdot \text{end}_1 \cdot \text{end}_2 \dots \text{end}_n$, where each $\text{end}_i$ is broadcast by the component simulating $\mathcal{A}_i$. In addition to transitions in Figures 7 and 8, $T$ also includes: $\{(q_{in}, !!a, q_{in}) \mid a \in A\} \cup \{(q_{in}, !!\$, q_{in})\}$. One can show that there exists a word $w = a_1 \dots a_n$ in the intersection of the $n$ automata iff there is an execution in which $t_f$ is taken infinitely often.

▶ **Theorem 5.4.** *RepCover is PSpace-hard.*

## References


1. K. R. Apt and D. C. Kozen. Limits for automatic verification of finite-state concurrent systems. *Inf. Process. Lett.*, 22(6):307–309, 1986.
2. Peter Chini, Roland Meyer, and Prakash Saivasan. Liveness in broadcast networks. *Computing*, 104(10):2203–2223, 2022. URL: `https://doi.org/10.1007/s00607-021-00986-y`, `doi:10.1007/S00607-021-00986-Y`.
3. Wojciech Czerwinski and Lukasz Orlikowski. Reachability in vector addition systems is ackermann-complete. In *62nd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2021, Denver, CO, USA, February 7-10, 2022*, pages 1229–1240. IEEE, 2021. `doi:10.1109/FOCS52979.2021.00120`.
4. Giorgio Delzanno, Jean-François Raskin, and Laurent Van Begin. Towards the automated verification of multithreaded java programs. In *TACAS 2002*, volume 2280 of *LNCS*, pages 173–187. Springer, 2002. `doi:10.1007/3-540-46002-0\_13`.
5. Giorgio Delzanno, Arnaud Sangnier, and Gianluigi Zavattaro. Parameterized verification of ad hoc networks. In Paul Gastin and François Laroussinie, editors, *CONCUR 2010 - Concurrency Theory*, pages 313–327, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.
6. E.A. Emerson and K.S. Namjoshi. On model checking for non-deterministic infinite-state systems. In *Proceedings. Thirteenth Annual IEEE Symposium on Logic in Computer Science (LICS)*, pages 70–80, 1998. `doi:10.1109/LICS.1998.705644`.
7. J. Esparza, P. Ganty, and R. Majumdar. Parameterized verification of asynchronous shared-memory systems. In *CAV'13*, volume 8044 of *LNCS*, pages 124–140. Springer-Verlag, 2013.
8. Javier Esparza, Alain Finkel, and Richard Mayr. On the verification of broadcast protocols. In *14th Annual IEEE Symposium on Logic in Computer Science, Trento, Italy, July 2-5, 1999*, pages 352–359. IEEE Computer Society, 1999. `doi:10.1109/LICS.1999.782630`.
9. S. M. German and A. P. Sistla. Reasoning about systems with many processes. *Journal of the ACM*, 39(3):675–735, 1992.
10. Lucie Guillou, Arnaud Sangnier, and Nathalie Sznajder. Safety analysis of parameterised networks with non-blocking rendez-vous. In *CONCUR'23*, volume 279 of *LIPIcs*, pages 7:1–7:17. loss Dagstuhl - Leibniz-Zentrum für Informatik, 2023.
11. Lucie Guillou, Arnaud Sangnier, and Nathalie Sznajder. Safety verification of wait-only non-blocking broadcast protocols. In *Application and Theory of Petri Nets and Concurrency - 45th International Conference, PETRI NETS 2024*, volume 14628 of *Lecture Notes in Computer Science*, pages 291–311. Springer, 2024. `doi:10.1007/978-3-031-61433-0\_14`.
12. Lucie Guillou, Arnaud Sangnier, and Nathalie Sznajder. Wait-only broadcast protocols are easier to verify, 2025. URL: `https://arxiv.org/abs/2506.22144`, `arXiv:2506.22144`.
13. Peter Habermehl. On the complexity of the linear-time mu -calculus for petri-nets. In Pierre Azéma and Gianfranco Balbo, editors, *Application and Theory of Petri Nets 1997, 18th International Conference, ICATPN '97, Toulouse, France, June 23-27, 1997, Proceedings*, volume 1248 of *Lecture Notes in Computer Science*, pages 102–116. Springer, 1997. `doi:10.1007/3-540-63139-9\_32`.

679 **14** John E. Hopcroft and Jean-Jacques Pansiot. On the reachability problem for 5-dimensional
680 vector addition systems. *Theor. Comput. Sci.*, 8:135–159, 1979. doi:10.1016/0304-3975(79)
681 90041-0.
682 **15** Dexter Kozen. Lower bounds for natural proof systems. In *18th Annual Symposium on*
683 *Foundations of Computer Science, Providence, Rhode Island, USA, 31 October - 1 November*
684 *1977*, pages 254–266. IEEE Computer Society, 1977. doi:10.1109/SFCS.1977.16.
685 **16** Jérôme Leroux. Vector addition system reversible reachability problem. In Joost-Pieter Katoen
686 and Barbara König, editors, *CONCUR 2011 - Concurrency Theory - 22nd International*
687 *Conference, CONCUR 2011, Aachen, Germany, September 6-9, 2011. Proceedings*, volume
688 6901 of *Lecture Notes in Computer Science*, pages 327–341. Springer, 2011. doi:10.1007/
689 978-3-642-23217-6_22.
690 **17** Jérôme Leroux. The reachability problem for petri nets is not primitive recursive. In *62nd IEEE*
691 *Annual Symposium on Foundations of Computer Science, FOCS 2021, Denver, CO, USA,*
692 *February 7-10, 2022*, pages 1241–1252. IEEE, 2021. doi:10.1109/FOCS52979.2021.00121.
693 **18** Jérôme Leroux and Sylvain Schmitz. Reachability in vector addition systems is primitive-
694 recursive in fixed dimension. In *34th Annual ACM/IEEE Symposium on Logic in Computer*
695 *Science, LICS 2019, Vancouver, BC, Canada, June 24-27, 2019*, pages 1–13. IEEE, 2019.
696 doi:10.1109/LICS.2019.8785796.
697 **19** R.J. Lipton. *The reachability problem requires exponential space*. Research report (Yale Uni-
698 versity. Department of Computer Science). Department of Computer Science, Yale University,
699 1976. URL: https://books.google.fr/books?id=7iSbGwAACAAJ.
700 **20** S. Schmitz and P. Schnoebelen. The power of well-structured systems. In *CONCUR'13*,
701 volume 8052 of *Lecture Notes in Computer Science*, pages 5–24. Springer, 2013.
702 **21** Rüdiger Valk. Self-modifying nets, a natural extension of petri nets. In *ICALP'78*, volume 62
703 of *LNCS*, pages 464–476. Springer, 1978.

# A Synchro is undecidable

In order to prove undecidability of Synchro, we reduce the halting problem of a 2-counter machine.

## 2-counter machines

A 2-counter machine is a tuple $M = (S, s_{in}, s_f, \mathsf{x}_1, \mathsf{x}_2, \Delta)$ such that $S$ is a finite set of states, $s_{in} \in S$ is the initial state, $s_f \in S$ is the final state, $\mathsf{x}_1$ and $\mathsf{x}_2$ are two counters that take their values in $\mathbb{N}$, and $\Delta \subseteq S \times \mathrm{Op} \times S$ is a finite set of transitions, with $\mathrm{Op} = \{\mathsf{x}++, \mathsf{x}--, \mathsf{x}=0 \mid \mathsf{x} \in \{\mathsf{x}_1, \mathsf{x}_2\}\}$. A configuration of the machine is a tuple $(s, x_1, x_2)$ with $s \in S$, and $x_1 \in \mathbb{N}$ (resp. $x_2 \in \mathbb{N}$) is the value of counter $\mathsf{x}_1$ (resp. $\mathsf{x}_2$).

Consider a transition $\delta = (s, \mathrm{op}, s') \in \Delta$, then for $x_1, x_2 \in \mathbb{N}$, we have:

- $(s, x_1, x_2) \overset{\delta}{\rightsquigarrow} (s', x_1 + 1, x_2)$ iff $\mathrm{op} = \mathsf{x}_1 ++$;
- $(s, x_1, x_2) \overset{\delta}{\rightsquigarrow} (s', x_1, x_2 + 1)$ iff $\mathrm{op} = \mathsf{x}_2 ++$;
- $(s, x_1, x_2) \overset{\delta}{\rightsquigarrow} (s', x_1 - 1, x_2)$ iff $x_1 > 0$ and $\mathrm{op} = \mathsf{x}_1 --$;
- $(s, x_1, x_2) \overset{\delta}{\rightsquigarrow} (s', x_1, x_2 - 1)$ iff $x_2 > 0$ and $\mathrm{op} = \mathsf{x}_2 --$;
- $(s, x_1, x_2) \overset{\delta}{\rightsquigarrow} (s', x_1, x_2)$ iff $x_1 = 0$ and $\mathrm{op} = (\mathsf{x}_1 = 0)$;
- $(s, x_1, x_2) \overset{\delta}{\rightsquigarrow} (s', x_1, x_2)$ iff $x_2 = 0$ and $\mathrm{op} = (\mathsf{x}_2 = 0)$;

We let $\Delta_{\mathrm{test}} = \{(s, \mathrm{op}, s') \in \Delta \mid \mathrm{op} \in \{\mathsf{x}_1 = 0, \mathsf{x}_2 = 0\}\} \subseteq \Delta$. Given two configurations $(s, x_1, x_2)$ and $(s', x_1', x_2')$, we write $(s, x_1, x_2) \rightsquigarrow (s', x_1', x_2')$ whenever there exists a transition $\delta \in \Delta$ such that $(s, x_1, x_2) \overset{\delta}{\rightsquigarrow} (s', x_1', x_2')$. Finally we denote $\rightsquigarrow^*$ the reflexive and transitive closure of $\rightsquigarrow$. We assume that $\Delta$ is deterministic, and that $s_f$ has no outgoing transition. The halting problem asks whether $(s_{in}, 0, 0) \rightsquigarrow^* (s_f, 0, 0)$ and it is undecidable.

## Simulation.

Given a machine $M = (S, s_{in}, s_f, \mathsf{x}_1, \mathsf{x}_2, \Delta)$, we define a protocol $P$ depicted in Figure 9, where $P_M$ is defined as follows: $P_M = (S \cup \{☹\}, T_M)$ with $T_M = \{(s, ?\mathrm{op}, s') \mid (s, \mathrm{op}, s') \in \Delta \setminus \Delta_{\mathrm{test}}\} \cup \{(s, !!\mathsf{x}{=}0, s') \mid (s, \mathsf{x}{=}0, s') \in \Delta_{\mathrm{test}}\} \cup \{(s, ?\mathrm{op}, ☹) \mid s \in S, \mathrm{op} \in \{\mathsf{x}++, \mathsf{x}-- \mid \mathsf{x} \in \{\mathsf{x}_1, \mathsf{x}_2\}\}\}$.

If $(s_{in}, 0, 0) \rightsquigarrow^* (s_f, 0, 0)$, there is an execution of the protocol that gathers all the processes in $s_f$. The number of processes needed is $1 + \max(x_1 + x_2)$ where $\max(x_1 + x_2)$ denotes the maximum value of the sum of the counters during the execution of $M$. Initially, a single process (the leader), broadcasts $\ell$, and goes to $s_{in}$, while all the others go to the state 0. From that point onward, the configurations of the 2-counter machines are encoded in a the protocol's configurations as follows: the state of the leader process determines the location of $M$, and the number of processes in state $x_1$ (resp. $x_2$) represents the value of counter $\mathsf{x}_1$ (resp. $\mathsf{x}_2$).

From a location $s$, if the transition taken in the execution of $M$ belongs to $\Delta_{\mathrm{test}}$, the leader process broadcasts the corresponding message $!!\mathsf{x}_i = 0$. Since the counter $\mathsf{x}_i$ is zero at that moment, no process is in state $x_i$, allowing the simulation to continue. If the transition involves an increment (respectively, a decrement) of counter $\mathsf{x}_i$, then, by our choice of the number of processes, there are some processes in state 0 (respectively, in state $x_i$). One such process broadcasts the message $!!\mathsf{x}_i ++$ (respectively, $!!\mathsf{x}_i --$), which causes the leader to go to the next location. The protocol configuration remains equivalent to the configuration of $M$. Since $(s_f, 0, 0)$ is reached in $M$, this simulation ensures that the leader process eventually reaches state $s_f$, while all other processes remain in state 0. At that moment, the leader broadcasts $!!d$, bringing all processes to $s_f$.

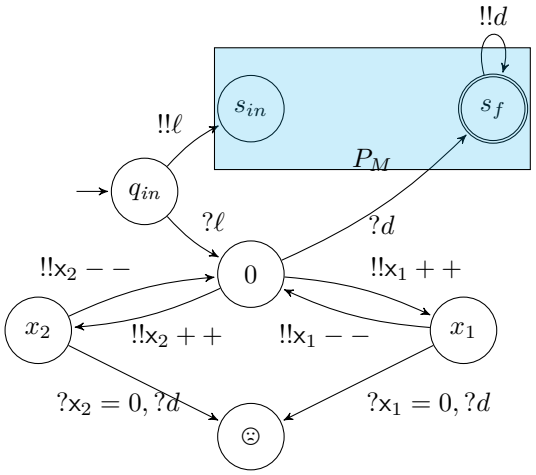


**Figure 9** Protocol $P$ associated to a machine $M = (S, s_{in}, s_f, \mathsf{x}_1, \mathsf{x}_2, \Delta)$.

Conversely, if there exists an execution of the protocol that gathers all processes in $s_f$, then at some point, one process must have broadcast $!!\ell$, after which all other processes must have reached state 0. From this point, if any process deviates from the intended behavior—such as a counter sending an incorrect operation to the leader or the leader sending a zero-test message when the corresponding counter is not empty—an erroneous message is received by some processes, leading them to the sink state ☹, from which it is impossible to gather all processes in $s_f$. When the leader process eventually reaches $s_f$ (which must happen since the execution gathers all processes in $s_f$), the only way to ensure that all other processes join is by broadcasting $!!d$. If any process is not in state 0 at that moment, it will be sent to the sink state ☹. Consequently, the corresponding execution in the 2-counter machine must be correct and must reach $(s_f, 0, 0)$.

## B Proof of Lemma 3.1

**Proof.** Let $\rho \in \Lambda \cup \Lambda_\omega$ be an execution of $P$ such that there exist $e_1, e_2 \in [1, \#\mathbf{proc}(\rho)]$ and $0 \leq j_0 < |\rho|$ with $\rho_{j_0}(e_1) = \rho_{j_0}(e_2) \in Q_W$ and $\mathbf{na}(\rho, j_0, e_1) = (q, j_1)$ and $\mathbf{na}(\rho, j_0, e_2) = (q, j_2)$ with $j_1 \leq j_2 < |\rho|$.

We define the sequence of configurations $\rho' = \rho'_{j_0+1} \cdots \rho'_{j_2}$ as follows:

- for all $j_0 < k \leq j_1$, $\rho'_k(e_2) = \rho_k(e_1)$ and for all $e \neq e_2$, $\rho'_k(e) = \rho_k(e)$;
- for all $j_1 < k \leq j_2$, $\rho'_k(e_2) = q$ and for all $e \neq e_2$, $\rho'_k(e) = \rho_k(e)$.

Observe that $\rho_{j_2}(e_2) = q$ as $\mathbf{na}(\rho, j_0, e_2) = (q, j_2)$. Hence, $\rho'_{j_2} = \rho_{j_2}$. Hence, it remains to prove that $\rho_{j_0} \to \rho'_{j_0+1} \to \cdots \to \rho'_{j_2}$ in order to prove that $\rho_{\leq j_0} \cdot \rho' \cdot \rho_{\geq j_2+1}$ is an execution.

We do so by induction: we let $\rho'_{j_0} = \rho_{j_0}$ and we prove by induction on $j_0 \leq k \leq j_2$ that $\rho'_{j_0} \to \rho'_{j_0+1} \to \cdots \to \rho'_k$.

For $k = j_0$, there is nothing to do. Let $j_0 \leq k < j_2$ and assume that $\rho'_{j_0} \to \rho'_{j_0+1} \to \cdots \to \rho'_k$. Let us prove that $\rho'_k \to \rho'_{k+1}$. To do so, let $\rho_k \xrightarrow{e_0,t} \rho_{k+1}$ with $t = (q, !!a, q')$. We distinguish between three cases:

- Case $k < j_1$. First observe that $k < j_1 = \mathbf{na\text{-}index}(\rho, j_0, e_1) \leq \mathbf{na\text{-}index}(\rho, j_0, e_2) = j_2$, hence $\rho_k(e_1) \in Q_W$, $\rho_k(e_2) \in Q_W$ and so $e_0 \neq e_1$ and $e_0 \neq e_2$.

Observe that we are in the case where $\rho'_k(e_2) = \rho_k(e_1)$ and $\rho'_{k+1}(e_2) = \rho_{k+1}(e_1)$. As $\rho_k \to \rho_{k+1}$, either $(\rho'_k(e_2), ?a, \rho'_{k+1}(e_2)) = (\rho_k(e_1), ?a, \rho_{k+1}(e_1)) \in T$ or $a \notin R(\rho_k(e_1)) = R(\rho'_k(e_2))$ and $\rho'_{k+1}(e_2) = \rho_{k+1}(e_1) = \rho_k(e_1) = \rho'_k(e_2)$. Hence, as for all other process $e$, $\rho'_k(e) = \rho_k(e)$ and $\rho'_{k+1}(e) = \rho_{k+1}(e)$, it holds that $\rho'_k \xrightarrow{e_0,t} \rho'_{k+1}$.

- Case $k > j_1$. First observe that $k < j_2 = \mathbf{na\text{-}index}(\rho, j_0, e_2)$, hence $\rho_k(e_2) \in Q_W$, and so $e_0 \neq e_2$.
  Observe that we are in the case where $\rho'_k(e_2) = \rho'_{k+1}(e_2) = q$. Recall that $q \in Q_A$. Hence, $a \notin R(q) = \emptyset$.
  As $\rho'_k(e) = \rho_k(e)$ and $\rho'_{k+1}(e) = \rho_{k+1}(e)$ for all other process $e$, it holds that $\rho'_k \xrightarrow{e,t} \rho'_{k+1}$.
  As in the previous case, we conclude that $\rho'_k \xrightarrow{e_0,t} \rho'_{k+1}$.
- Case $k = j_1$. In that case $\rho'_{j_1}(e_2) = \rho_{j_1}(e_1) = q$ and $\rho'_{j_1+1}(e_2) = q$. As $q \in Q_A$, $a \notin R(q) = \emptyset$ and so, as $\rho'_k(e) = \rho_k(e)$ and $\rho'_{k+1}(e) = \rho_{k+1}(e)$ for all other process $e \neq e_2$, it holds that $\rho'_k \xrightarrow{e_0,t} \rho'_{k+1}$.

◀

# C Proofs of Section 3.2

Let $\lambda = (\mathcal{S}, v)$ and $\lambda' = (\mathcal{S}', v')$ be two S-configurations and $t = (q, !!a, q') \in T$. When $\lambda \overset{\delta}{\rightsquigarrow\!\triangleright} \lambda'$ and $(\mathcal{S}, \delta, \mathcal{S}') \in \Delta_t$, we say that $\lambda \overset{t}{\rightsquigarrow\!\triangleright} \lambda'$.

**Proof of Lemma 3.8.** Let $\lambda \overset{\delta}{\rightsquigarrow\!\triangleright} \lambda'$ with $\lambda = (\mathcal{S}, v)$ and $\lambda' = (\mathcal{S}', v')$. Let $C \in [\![\lambda]\!]$ with $||C|| = n$ and $f : [1, n] \to \mathsf{X}$.

- If $\delta \in \Delta_e$, then $\mathcal{S} = \mathcal{S}'$ and let $(p, k) \notin \mathbf{lab}(\mathcal{S})$ such that $\delta(\mathsf{x}_p) = 1$ and $\delta(\mathsf{x}_{p,k}) = -1$. In that case, we let $C' = C$ and $C \to^* C'$. It remains to show that $C = C' \in [\![\lambda']\!]$. For that, we will build another function $f' : [1, n] \to \mathsf{X}$, that reflects the modification of counters $\mathsf{x}_{p,k}$ and $\mathsf{x}_p$. Since $\delta(\mathsf{x}_{p,k}) = -1$, $v(\mathsf{x}_{p,k}) > 0$ and hence $|f^{-1}(\mathsf{x}_{p,k})| > 0$. Thus, let $e \in f^{-1}(\mathsf{x}_{p,k})$. Since $(p, k) \notin \mathbf{lab}(\mathcal{S})$, by **CondImpl3**, $C(e) = C'(e) = p$. Then, we let $f'(e) = \mathsf{x}_p$, and for all other processes $e' \in [1, n]$, we let $f'(e) = f(e)$. Then, $|f'^{-1}(\mathsf{x}_p)| = |f^{-1}(\mathsf{x}_p)| + 1$, $|f'^{-1}(\mathsf{x}_{p,k})| = f^{-1}(\mathsf{x}_{p,k})| - 1$, and for all other counter $\mathsf{x}$, $|f'^{-1}(\mathsf{x})| = |f^{-1}(\mathsf{x})|$. Then, $v'(\mathsf{x}_p) = v(\mathsf{x}_p) + 1 = |f^{-1}(\mathsf{x}_p)| + 1 = |f'^{-1}(\mathsf{x}_p)|$, $v'(\mathsf{x}_{p,k}) = v(\mathsf{x}_{p,k}) - 1 = |f^{-1}(\mathsf{x}_{p,k})| - 1 = |f'^{-1}(\mathsf{x}_{p,k})|$, and for all other counter $v'(\mathsf{x}) = v(\mathsf{x}) = |f'^{-1}(\mathsf{x})|$. Since $C'(e) = p$, it is easy to see that $f'$ meets the requirements **CondImpl1**, **CondImpl2** and **CondImpl3**, hence $C' \in [\![\lambda']\!]$.
- Otherwise, $\delta \notin \Delta_e$ and let $t = (q, !!a, q')$ such that $\lambda \overset{t}{\rightsquigarrow\!\triangleright} \lambda'$. Since $\delta(\mathsf{x}_q) = -1$, $v(\mathsf{x}_q) > 0$ and hence there exists $e_q \in [1, n]$ such that $f(e_q) = \mathsf{x}_q$, and, by **CondImpl1**, $C(e_q) = q$. We build $C'$ as follows: $C'(e_q) = q'$. Then for all other $e \in [1, n]$, if $a \notin R(C(e))$, we let $C'(e) = C(e)$. Otherwise, $C(e) \in Q_W$. From the fact that $C \in [\![\lambda]\!]$, we deduce that $f(e) = \mathsf{x}_{(p,k)}$, with $(p, k) \in \mathbf{lab}(\mathcal{S})$, and let $S = (\mathsf{pr}, p, k) \in \mathcal{S}$. Now,
  - if $(p, k) \notin \mathbf{lab}(\mathcal{S}')$, it means that $S \overset{t}{\Rightarrow} \mathtt{Done}$, and we let $C'(e) = p$. By definition of $S \overset{t}{\Rightarrow} \mathtt{Done}$, we deduce that $(C(e), ?a, p) \in T$.
  - Otherwise, let $S' = (\mathsf{pr}', p, k) \in \mathcal{S}'$. By definition of the VASS, it means that either $S \overset{t}{\Rightarrow} S'$, or $S \overset{t,+q'}{\Longrightarrow} S'$. In both cases, there exists a configuration $C''$ such that $\overset{\bullet}{q} \oplus \overset{\bullet}{\mathsf{pr}} \xrightarrow{1,t} \overset{\bullet}{q'} \oplus C''$ and either $\mathbf{set}(C'') = \mathsf{pr}'$ or $\mathbf{set}(C'') \cup \{q'\} = \mathsf{pr}'$. Let $e'$ such that $\overset{\bullet}{q} \oplus \overset{\bullet}{\mathsf{pr}}(e') = C(e)$ (we will justify its existence just after), then, we define

$C'(e) = \dot{q'} \oplus C''(e')$. Observe that we know that such a process $e'$ exists, thanks to **CondImpl2**: since $e \in f^{-1}(\mathsf{x}_{p,k})$, $C(e) \in \mathsf{pr} \cup \{p\}$. Moreover, $C(e) \in Q_W$, hence $C(e) \in \mathsf{pr}$. Then we know that $(C(e), ?a, C'(e)) \in T$ or $C(e) = C'(e)$ and $a \notin R(C(e))$.

It is clear from the construction of $C'$ that $C \to C'$. We show now that $C' \in [\![\lambda']\!]$ by defining $f' : [1, n] \to \mathsf{X}$ as follows.

$$f'(e_q) = \begin{cases} \mathsf{x}_{q'} & \text{if } q' \in Q_A \\ \mathsf{x}_{p,k} & \text{if } q' \in Q_W, \text{ and } \mathbf{lab}(\mathcal{S}) = \mathbf{lab}(\mathcal{S}'), \text{ with } \mathsf{x}_{p,k} \text{ the unique counter such that } \delta(\mathsf{x}_{p,k}) = 1 \\ \mathsf{x}_{p,k} & \text{otherwise, with } (p,k) \in \mathbf{lab}(\mathcal{S}') \setminus \mathbf{lab}(\mathcal{S}) \end{cases}$$

$$f'(e) = f(e) \quad \text{for all other } e \in [1, n]$$

We have to prove **CondImpl1**, **CondImpl2** and **CondImpl3**, but also that for all $\mathsf{x} \in \mathsf{X}$, $|f'^{-1}(\mathsf{x})| = v'(\mathsf{x})$.

Let $p \in Q_A$ and $e \neq e_q$ such that $f'(e) = p = f(e)$ by construction. Then $C(e) = p$, and by definition of $C'$, $C'(e) = p$. If $e_q$ is such that $f'(e_q) = p$, it means that $p = q' \in Q_A$. In that case, $C'(e_q) = p$. In any case, **CondImpl1** is satisfied. Moreover, if $p \neq q'$, $v'(\mathsf{x}_p) = v(\mathsf{x}_p)$ and $f'^{-1}(\mathsf{x}_p) = f^{-1}(\mathsf{x}_p)$, hence $|f'^{-1}(\mathsf{x}_p)| = v'(\mathsf{x}_p)$. If $p = q'$, then $v'(\mathsf{x}_p) = v(\mathsf{x}_p) + 1$ and $f'^{-1}(\mathsf{x}_p) = f^{-1}(\mathsf{x}_p) \cup \{e_q\}$ thus $|f'^{-1}(\mathsf{x}_p)| = v'(\mathsf{x}_p)$.

Let $(p,k)$ such that $S' = (\mathsf{pr}', p, k) \in \mathcal{S}'$ with $(p,k) \in \mathbf{lab}(\mathcal{S})$. Let $e \in f'^{-1}(\mathsf{x}_{p,k})$. If $e \neq e_q$, then $e \in f^{-1}(\mathsf{x}_{p,k})$. Let $S = (\mathsf{pr}, p, k) \in \mathcal{S}$. Then $C(e) \in \mathsf{pr} \cup \{p\}$. By construction of $C'$, either $C(e) = p = C'(e)$, or $C(e) \in \mathsf{pr}$ and $C'(e) \in \mathsf{pr}'$. If $e = e_q$, by definition of $f'$, $q' \in Q_W$, $\mathbf{lab}(\mathcal{S}) = \mathbf{lab}(\mathcal{S}')$, and $\delta(\mathsf{x}_{p,k}) = 1$, and, by definition of the VASS, this means that $S \xrightarrow{t,+q'} S'$, hence $q' \in \mathsf{pr}'$. Hence, $C'(e) = q' \in \mathsf{pr}'$. Let now $(p,k)$ such that $S' = (\mathsf{pr}', p, k) \in \mathcal{S}'$ with $(p,k) \notin \mathbf{lab}(\mathcal{S})$. Let $e \in f'^{-1}(\mathsf{x}_{p,k})$ such that $e \neq e_q$. In that case, $e \in f^{-1}(\mathsf{x}_{p,k})$. Since $(p,k) \notin \mathbf{lab}(\mathcal{S})$, by **CondImpl3**, $C(e) = p \in Q_A$, and then $C'(e) = p \in \mathsf{pr}' \cup \{p\}$. If $e = e_q$, $C'(e) = q'$, and by definition of the VASS, $S' = (\{q'\}, p, k)$.

Then **CondImpl2** is met. Moreover, if $f'(e_q) = \mathsf{x}_{p,k}$ then $|f'^{-1}(\mathsf{x}_{p,k})| = |f^{-1}(\mathsf{x}_{p,k})| + 1$, and in that case, $v'(\mathsf{x}_{p,k}) = v(\mathsf{x}_{p,k}) + 1$. Otherwise, $|f'^{-1}(\mathsf{x}_{p,k})| = |f^{-1}(\mathsf{x}_{p,k})| = v(\mathsf{x}_{p,k}) = v'(\mathsf{x}_{p,k})$.

Finally, let $(p,k) \notin \mathbf{lab}(\mathcal{S}')$, and let $e \in f'^{-1}(\mathsf{x}_{p,k})$. By definition of $f'$, $e \in f^{-1}(\mathsf{x}_{p,k})$. If $(p,k) \notin \mathbf{lab}(\mathcal{S})$, then by **CondImpl3** on $C$, $C(e) = p \in Q_A$, and then $C'(e) = p$. Otherwise, let $S = (\mathsf{pr}, p, k) \in \mathcal{S}$, we must have $S \overset{t}{\Rightarrow} \texttt{Done}$. By construction, $C'(e) = p$. Then **CondImpl3** is satisfied by $C'$. Moreover, $v(\mathsf{x}_{p,k}) = v'(\mathsf{x}_{p,k})$ and $|f'^{-1}(\mathsf{x}_{p,k})| = |f^{-1}(\mathsf{x}_{p,k})|$.

We have hence proved that $C' \in [\![\lambda']\!]$.

◀

## D Proofs of Section 3.2

**Proof of Lemma 3.10.** Let $C = \rho_i$ and let $\lambda_i = (\mathcal{S}, v) \in \mathbf{repr}(\rho, i)$, then

- **CondRepr1** for all $q \in Q_A$, $v(\mathsf{x}_q) = |C^{-1}(q)|$;

and for all $q_a \in Q_A$, there is an injective function $r_{q_a} : \mathbf{na\text{-}index\text{-}set}(\rho, i, q_a) \to [1, |Q_W| + 1]$ s.t.:

- **CondRepr2** $\mathcal{S} = \{(C(E^{\rho,i}_{q_a,j}), q_a, r_{q_a}(j)) \mid q_a \in Q_A, j \in \mathbf{na\text{-}index\text{-}set}(\rho, i, q_a)\}$,and,

- **CondRepr3** for all $q_a \in Q_A$, we have $v(\mathsf{x}_{q_a,r_{q_a}(j)}) = |E^{\rho,i}_{q_a,j}|$ for all $j \in \mathbf{na\text{-}index\text{-}set}(\rho,i,q_a)$ and $v(\mathsf{x}_{q_a,k}) = 0$ for all $k \notin r_{q_a}(\mathbf{na\text{-}index\text{-}set}(\rho,i,q_a))$.

Let $t = (q, !!m, q')$ and $e_q \in [1, \#\mathbf{proc}(\rho)]$ such that $C \xrightarrow{e_q,t} C_{i+1}$. We start by building an $S$-configuration $\lambda' = (\mathcal{S}', v')$ such that $\lambda_i \overset{t}{\rightsquigarrow} \lambda'$. We will then show that an appropriate sequence of transitions from $\Delta_e$ allows to go from $\lambda'$ to a configuration $\lambda_{i+1} \in \mathbf{repr}(\rho, i+1)$.

We state first some remarks about $E^{\rho,i}_{q_a,j}$, for $q_a \in Q_A$ and $j \in \mathbf{na\text{-}index\text{-}set}(\rho,i,q_a)$. Observe that, since $E^{\rho,i}_{q_a,j} = \{e \in [1, \#\mathbf{proc}(\rho)] \mid C(e) \subseteq Q_W \text{ and } \mathbf{na}(\rho,i,e) = (q_a,j)\}$, if $C_{i+1}(e) \in Q_A$ for some $e \in E^{\rho,i}_{q_a,j}$, then it means that $j = i+1$ and $C_{i+1}(e) = q_a$ for all $e \in E^{\rho,i}_{q_a,j}$. Hence $C_{i+1}(E^{\rho,i}_{q_a,j}) \subseteq Q_W$ or $C_{i+1}(E^{\rho,i}_{q_a,j}) = \{q_a\}$.

Observe also that if $j \in \mathbf{na\text{-}index\text{-}set}(\rho,i,q_a)$:

$$E^{\rho,i+1}_{q_a,j} = \begin{cases} \emptyset & \text{if } j = i+1 \\ E^{\rho,i}_{q_a,j} \cup \{e_q\} & \text{if } q' \in Q_W \text{ and } \mathbf{na}(\rho,i+1,e_q) = (q_a,j) \\ E^{\rho,i}_{q_a,j} & \text{otherwise.} \end{cases}$$

Moreover, if $q' \in Q_W$, and $\mathbf{na}(\rho,i,e_q) = (q_a,j)$ with $j \notin \mathbf{na\text{-}index\text{-}set}(\rho,i,q_a)$, then $E^{\rho,i}_{q_a,j} = \emptyset$ hence $E^{\rho,i+1}_{q_a,j} = \{e_q\}$.

### Definition of $\lambda'$.

For all $S = (\mathsf{pr}, q_a, k) \in \mathcal{S}$, we let $j = r^{-1}_{q_a}(k)$ and we define

$$S^{+1} = \begin{cases} (C_{i+1}(E^{\rho,i}_{q_a,j}), q_a, k) & \text{if } C_{i+1}(E^{\rho,i}_{q_a,j}) \subseteq Q_W \\ \mathtt{Done} & \text{otherwise.} \end{cases}$$

Recall that $t = (q, !!m, q')$. $S^{+1}$ represents the set of states reached by the states in the print of $S$ after the sending of the message $m$ in $\rho$. We then let $\mathcal{S}_0 = \{S^{+1} \mid S \in \mathcal{S}\} \setminus \{\mathtt{Done}\}$.

We first show that $\mathcal{S}_0$ is coherent: let $S_1 = (\mathsf{pr}_1, q_a, k_1), S_2 = (\mathsf{pr}_2, q_a, k_2) \in \mathcal{S}$ such that ${S_1}^{+1} = (\mathsf{pr}'_1, q_a, k_1)$ and ${S_2}^{+1} = (\mathsf{pr}'_2, q_a, k_2)$ are two distinct summaries in $\mathcal{S}_0$. Since $\mathcal{S}$ is coherent, we know that $k_1 \neq k_2$. Assume that there is some $p \in \mathsf{pr}'_1 \cap \mathsf{pr}'_2$. It means that there exist $e_1, e_2 \in [1, \#\mathbf{proc}(\rho)]$ such that $C_{i+1}(e_1) = p = C_{i+1}(e_2)$, and $e_1 \in E^{\rho,i}_{q_a,r^{-1}_{q_a}(k_1)}$ and $e_2 \in E^{\rho,i}_{q_a,r^{-1}_{q_a}(k_2)}$. Hence by definition, $\mathbf{na}(\rho,i,e_1) = (q_a, r^{-1}_{q_a}(k_1)) = \mathbf{na}(\rho,i+1,e_1)$ and $\mathbf{na}(\rho,i,e_2) = (q_a, r^{-1}_{q_a}(k_2)) = \mathbf{na}(\rho,i+1,e_2)$. Since $\rho$ is well-formed, we have that $r^{-1}_{q_a}(k_1) = r^{-1}_{q_a}(k_2)$, and hence, $k_1 = k_2$, which is a contradiction. Then for any two distinct summaries ${S_1}^{+1} = (\mathsf{pr}'_1, q_a, k_1)$ and ${S_2}^{+1} = (\mathsf{pr}'_2, q_a, k_2)$, then $k_1 \neq k_2$ and $\mathsf{pr}'_1 \cap \mathsf{pr}'_2 = \emptyset$.

Now that we have defined $\mathcal{S}_0$, which captures the behavior of the processes that were in waiting states when the transition $t$ has been taken, we define $\mathcal{S}'$ that incorporates the process $e_q$ that arrives in state $q'$, along with the new valuation $v'$. In particular, if $q' \in Q_W$, then $e_q$ reaches a zone of waiting states and must enter one of the summaries. Observe first that $C(e_q) = q$, then $v(\mathsf{x}_q) = |C^{-1}(q)| > 0$, then we let $v'(\mathsf{x}_q) = v(\mathsf{x}_q) - 1$.

- If $q' \in Q_A$, $\mathcal{S}' = \mathcal{S}_0$ and $v'(\mathsf{x}_{q'}) = v(\mathsf{x}_{q'}) + 1$. For all other counter $\mathsf{x}$, $v'(\mathsf{x}) = v(\mathsf{x})$.
- If $q' \in Q_W$, let $\mathbf{na}(\rho,i+1,e_q) = (p,j)$.

1. If $j \in \mathbf{na\text{-}index\text{-}set}(\rho,i,p)$, then $e_q$ will join an existing summary $S = (\mathsf{pr}, p, r_p(j)) \in \mathcal{S}$. Since $\mathbf{na}(\rho,i+1,e_q) = (p,j)$ and $q' \in Q_W$, $j \neq i+1$ and $C_{i+1}(E^{\rho,i}_{q_a,j}) \subseteq Q_W$. Hence, $S^{+1} \neq \mathtt{Done}$. We then let $\mathcal{S}' = (\mathcal{S}_0 \setminus \{S^{+1}\}) \cup \{(\mathbf{print}(S^{+1}) \cup \{q'\}, p, r_p(j))\}$. Then $v'(\mathsf{x}_{p,r_p(j)}) = v(\mathsf{x}_{p,r_p(j)}) + 1$ and $v'(\mathsf{x}) = v(\mathsf{x})$ for all other counter $\mathsf{x}$.

2. If $j \notin \mathbf{na\text{-}index\text{-}set}(\rho, i, p)$, then $e_q$ will enter a new summary. Observe first that $\mathbf{na\text{-}index\text{-}set}(\rho, i{+}1, p) = (\mathbf{na\text{-}index\text{-}set}(\rho, i, p) \cup \{j\}) \setminus \{i{+}1\}$. So $|\mathbf{na\text{-}index\text{-}set}(\rho, i, p)| \leq |\mathbf{na\text{-}index\text{-}set}(\rho, i+1, p)| \leq |Q_W| < 1 + |Q_W|$, and there exists $k \in [1, |Q_W| + 1]$ such that $k \notin r_p(\mathbf{na\text{-}index\text{-}set}(\rho, i, p))$. This means that $(p, k)$ is a label that is not used in the summaries of $\mathcal{S}_0$ and we let $\mathcal{S}' = \mathcal{S}_0 \cup \{(\{q'\}, p, k)\}$. Now, $v'(\mathsf{x}_{p,k}) = v(\mathsf{x}_{p,k}) + 1$, and for all other counter $\mathsf{x}$, $v'(\mathsf{x}) = v(\mathsf{x})$.

**Proof that $\lambda' = (\mathcal{S}', v')$ is an $S$-configuration.**

To prove that $\lambda'$ is an $S$-configuration, we need to show that $\mathcal{S}'$ is coherent. The definition of $\mathcal{S}'$ depends on the belonging of $q'$ to $Q_A$ or $Q_W$.

- If $q' \in Q_A$, then $\mathcal{S}' = \mathcal{S}_0$, and we have proved above that $\mathcal{S}_0$ is coherent.
- if $q' \in Q_W$, let $S_1 = (\mathsf{pr}_1, q_a, k_1)$ and $S_2 = (\mathsf{pr}_2, q_a, k_2) \in \mathcal{S}_1$, for some $q_a \in Q_A$. If $S_1, S_2 \in \mathcal{S}_0$, by coherence of $\mathcal{S}_0$, we have that $k_1 \neq k_2$ and $\mathsf{pr}_1 \cap \mathsf{pr}_2 = \emptyset$. Otherwise, assume that $S_2 \in \mathcal{S}' \setminus \mathcal{S}_0$. Depending on whether $e_q$ has joined an existing summary or a new one, $S_2 = (\mathsf{pr} \cup \{q'\}, p, k_2)$ with $\mathsf{pr}$ empty or not. Then $S_1 = (\mathsf{pr}_1, p, k_1)$ and if $\mathsf{pr} \neq \emptyset$, $(\mathsf{pr}, p, k_2) \in \mathcal{S}_0$. By coherence of $\mathcal{S}_0$, $k_1 \neq k_2$ and $\mathsf{pr}_1 \cap \mathsf{pr} = \emptyset$. Moreover, if $S_2 = (\{q'\}, p, k_2)$, by construction, $k_2 \neq k_1$. Assume now that there exists $p' \in \mathsf{pr}_1 \cap \mathsf{pr}_2$. Then $p' = q'$, and there exists $e_1 \in [1, \#\mathbf{proc}(\rho)]$ such that $C_{i+1}(e_1) = q'$ and $\mathbf{na}(\rho, i, e_1) = (p, r_p^{-1}(k_1))$. But $\mathbf{na}(\rho, i, e_q) = (p, j)$. Since $\rho$ is well-formed, $j = r_p^{-1}(k_1)$, which is impossible because $k_2 \neq k_1$. Hence $\mathsf{pr}_1 \cap \mathsf{pr}_2 = \emptyset$, and $\mathcal{S}_1$ is coherent.

**Proof that $\lambda_i \overset{t}{\rightsquigarrow} \lambda'$.**

We first show that for all $S \in \mathcal{S}$, $S \overset{t}{\Rightarrow} S^{+1}$. Let $S = (\mathsf{pr}, q_a, k) \in \mathcal{S}$ with $j' = r_{q_a}^{-1}(k)$. By construction, $\mathsf{pr} = C(E^{\rho,i}_{q_a,j'})$. Assume now that there exist $e_1, e_2 \in E^{\rho,i}_{q_a,j'}$ such that $C(e_1) = C(e_2)$. Since $\rho$ is well-defined, $C_{i+1}(e_1) = C_{i+1}(e_2)$. Then, for all $s \in \mathsf{pr}$, there exists a unique $s^{+1} \in Q$ such that $C_{i+1}(e) = s^{+1}$ for all $e \in C_i^{-1}(s)$. Moreover, $C_{i+1}(E^{\rho,i}_{q_a,j'}) = \{s^{+1} \mid s \in \mathsf{pr}\}$. We define a configuration $C'$ as follows. Let $e \in [1, ||\overset{\bullet}{\mathsf{pr}}||]$ such that $\overset{\bullet}{\mathsf{pr}}(e) = p_e \in \mathsf{pr}$. Then $C'(e) = {p_e}^{+1}$. Hence $\mathbf{set}(C') = \{s^{+1} \mid s \in \mathsf{pr}\} = C_{i+1}(E^{\rho,i}_{q_a,j'})$. We show now that $\overset{\bullet}{q} \oplus 1, \overset{\bullet}{\mathsf{pr}} \overset{t}{\to} \overset{\bullet}{q'} \oplus C'$. Let $e \in [1, ||\overset{\bullet}{\mathsf{pr}}||]$ such that $\overset{\bullet}{\mathsf{pr}}(e) = p_e$, then there exists $e' \in E^{\rho,i}_{q_a,j'}$ such that $C(e') = p_e$ and ${p_e}^{+1} = C_{i+1}(e')$. Since $C \overset{t}{\to} C_{i+1}$, then either $(p_e, ?m, {p_e}^{+1}) \in T$ or $m \notin R(p_e)$ and $p_e = {p_e}^{+1}$, and $\overset{\bullet}{q} \oplus \overset{\bullet}{\mathsf{pr}} \overset{1,t}{\longrightarrow} \overset{\bullet}{q'} \oplus C'$. We have already established that $\mathbf{set}(C') = C_{i+1}(E^{\rho,i}_{q_a,j'}) \subseteq Q_W$ or $\mathbf{set}(C') = C_{i+1}(E^{\rho,i}_{q_a,j'}) = \{q_a\}$. In the latter case, $S^{+1} = \mathtt{Done}$. In the former case, $S^{+1} = (C_{i+1}(E^i_{q_a,j'}), q_a, k)$ and $S \overset{t}{\Rightarrow} S^{+1}$. In any case, we have that $S \overset{t}{\Rightarrow} S^{+1}$.

Now we show that there exists $\delta$ such that $\lambda_i \overset{\delta}{\rightsquigarrow} \lambda'$ and $(\mathcal{S}, \delta, \mathcal{S}') \in \Delta_t$.

- If $q' \in Q_A$, we must have $\delta(\mathsf{x}_q) = -1$, $\delta(\mathsf{x}_{q'}) = 1$ and for all other counter $\mathsf{x}$, $\delta(\mathsf{x}) = 0$. Observe that we have $v'(\mathsf{x}_q) = v(\mathsf{x}_q) - 1$, $v'(\mathsf{x}_{q'}) = v(\mathsf{x}_{q'}) + 1$, and $v'(\mathsf{x}) = v(\mathsf{x})$ for all other counter $\mathsf{x}$. Moreover, $\mathcal{S}' = \mathcal{S}_0 = \{S^{+1} \mid S \in \mathcal{S}\} \setminus \{\mathtt{Done}\}$. Since $S \overset{t}{\Rightarrow} S^{+1}$ for all $S \in \mathcal{S}$, we have $(\mathcal{S}, \delta, \mathcal{S}') \in \Delta_t$ and $\lambda_i \overset{\delta}{\rightsquigarrow} \lambda'$.
- If $q' \in Q_W$ and there exists some $S = (\mathsf{pr}, p, r_p(j))$ such that $\mathcal{S}' = (\mathcal{S}_0 \setminus \{S^{+1}\}) \cup \{(\mathbf{print}(S^{+1}) \cup \{q'\}, p, r_p(j))\}$. Then $v'(\mathsf{x}_q) = v(\mathsf{x}_q) - 1$, $v'(\mathsf{x}_{p,r_p(j)}) = v(\mathsf{x}_{p,r_p(j)}) + 1$ and $v'(\mathsf{x}) = v(\mathsf{x})$ for all other counter $\mathsf{x}$. By definition, $S \overset{t,+q'}{\Longrightarrow} (\mathbf{print}(S^{+1}) \cup \{q'\}, p, r_p(j))$,

because $S \overset{t}{\Rightarrow} S^{+1}$. Then, with $\delta(\mathsf{x}_q) = -1$, $\delta(\mathsf{x}_{p,r_p(j)}) = 1$ and $\delta(\mathsf{x}) = 0$ for all other counter, we have $(\mathcal{S}, \delta, \mathcal{S}') \in \Delta_t$, and $\lambda_i \overset{\delta}{\rightsquigarrow} \lambda'$.

- If $q' \in Q_W$ and $\mathcal{S}' = \mathcal{S}_0 \cup \{(\{q'\}, p, k)\}$ for some $k \in [1, |Q_W| + 1]$. Then $(p, k) \notin \mathbf{lab}(\mathcal{S})$ by construction. Moreover, $v'(\mathsf{x}_q) = v(\mathsf{x}_q) - 1$, and $v'(\mathsf{x}_{p,k}) = 1$. For all other counter $\mathsf{x}$, $v'(\mathsf{x}) = v(\mathsf{x})$. For $\delta$ such that $\delta(\mathsf{x}_q) = -1$, $\delta(\mathsf{x}_{p,k}) = 1$ and $\delta(\mathsf{x}) = 0$ for all other $\mathsf{x}$, $(\mathcal{S}, \delta, \mathcal{S}') \in \Delta_t$ and $\lambda_i \overset{\delta}{\rightsquigarrow} \lambda'$.

Then $\lambda_i \overset{t}{\rightsquigarrow} \lambda'$.

### Definition of $\lambda_{i+1}$.

First observe that for all $q_a \in Q_A$, there is at most one $S = (\mathsf{pr}, q_a, k) \in \mathcal{S}$ such that $S^{+1} = \mathtt{Done}$. Indeed, suppose there exist $S_1 = (\mathsf{pr}_1, q_a, k_1), S_2 = (\mathsf{pr}_2, q_a, k_2) \in \mathcal{S}$ such that ${S_1}^{+1} = {S_2}^{+1} = \mathtt{Done}$. By definition, $\mathsf{pr}_1 = C(E^{\rho,i}_{q_a, r_{q_a}^{-1}(k_1)})$ and $\mathsf{pr}_2 = C(E^{\rho,i}_{q_a, r_{q_a}^{-1}(k_2)})$. Let $j_1 = r_{q_a}^{-1}(k_1)$ and $j_2 = r_{q_a}^{-1}(k_2)$. Then, since $C_{i+1}(E^{\rho,i}_{q_a,j_i}) \notin Q_W$ for $i \in \{1, 2\}$, necessarily $C_{i+1}(E^{\rho,i}_{q_a,j_1}) = C_{i+1}(E^{\rho,i}_{q_a,j_2}) = \{q_a\}$. This implies that $j_1 = j_2 = i + 1$ and then that $k_1 = k_2$, which contradicts the coherence of $\mathcal{S}$. This means that, there is at most one summary for each $q_a \in Q_A$ that disappears while going from $\mathcal{S}$ to $\mathcal{S}'$. This corresponds to a set of processes leaving a waiting zone and reaching the state $q_a$ together. We capture this new configuration of the protocol by transferring the value of the counter $\mathsf{x}_{(q_a, r_{q_a}(i+1))}$ into the counter $\mathsf{x}_{q_a}$.

So we define $v_{i+1}$ as follows. For all $q_a \in Q_A$,

- if $i + 1 \in \mathbf{na\text{-}index\text{-}set}(\rho, i, q_a)$, let $k = r_{q_a}(i + 1)$ and $v_{i+1}(\mathsf{x}_{q_a}) = v'(\mathsf{x}_{q_a}) + v'(\mathsf{x}_{q_a,k})$ and $v_{i+1}(\mathsf{x}_{q_a,k}) = 0$.
- Otherwise, $v_{i+1}(\mathsf{x}_{q_a}) = v'(\mathsf{x}_{q_a})$.

For all other counter $\mathsf{x}$, $v_{i+1}(\mathsf{x}) = v'(\mathsf{x})$.

We let $\lambda_{i+1} = (\mathcal{S}', v_{i+1})$.

### Proof that $\lambda_{i+1} \in \mathbf{repr}(\rho, i + 1)$.

We show that **CondRepr1** holds. Let $q_a \in Q_A$. Observe first that, for all $e \in [1, \#\mathbf{proc}(\rho)]$, $(C(e), ?m, q_a) \in T$ if and only if $e \in E^{\rho,i}_{q_a,i+1}$. Let $e_q \in [1, \#\mathbf{proc}(\rho)]$ such that $C \xrightarrow{e_q,t} C_{i+1}$. Then, $C(e_q) = q$, $C_{i+1}(e_q) = q'$. For all $e \in [1, \#\mathbf{proc}(\rho)]$, $C_{i+1}(e) = q_a$ if and only if ($C(e) = q_a$ and $q \neq q_a$, or $C(e) = q_a$ and $e \neq e_q$, or $(C(e), ?m, q_a) \in T$ and $\mathbf{na}(\rho, i, e) = (q_a, i + 1)$, or $q_a = q'$ and $e = e_q$). From this we can deduce that

$$C^{-1}_{i+1}(q_a) = \begin{cases} C^{-1}(q_a) \cup E^{\rho,i}_{q_a,i+1} & \text{if } q_a \neq q \text{ and } q_a \neq q' \\ C^{-1}(q_a) \cup \{e_q\} \cup E^{\rho,i}_{q_a,i+1} & \text{if } q_a = q' \\ (C^{-1}(q_a) \setminus \{e_q\}) \cup E^{\rho,i}_{q_a,i+1} & \text{if } q_a = q. \end{cases}$$

Since $C^{-1}(q_a)$ and $E^{\rho,i}_{q_a,i+1}$ are disjoint, and if $e_q \in C^{-1}(q) \setminus C^{-1}(q')$ (which is always true as $q \neq q'$ by hypothesis), we have that

$$|C^{-1}_{i+1}(q_a)| = \begin{cases} |C^{-1}(q_a)| + |E^{\rho,i}_{q_a,i+1}| & \text{if } q_a \neq q \text{ and } q_a \neq q' \\ |C^{-1}(q_a) + 1 + |E^{\rho,i}_{q_a,i+1}| & \text{if } q_a = q' \\ |C^{-1}(q_a)| - 1 + |E^{\rho,i}_{q_a,i+1}| & \text{if } q_a = q. \end{cases}$$

Since $\lambda = (\mathcal{S}, v) \in \mathbf{repr}(C, i)$, $v(\mathsf{x}_{q_a}) = |C^{-1}(q_a)|$ and, by definition of $v'$, we have that

$$|C_{i+1}^{-1}(q_a)| = v'(\mathsf{x}_{q_a}) + |E_{q_a,i+1}^{\rho,i}|$$

Now, if $i+1 \notin \mathbf{na\text{-}index\text{-}set}(\rho, i, q_a)$, it means that $E_{q_a,i+1}^{\rho,i} = \emptyset$. In that case, by definition of $v_{i+1}$, $v_{i+1}(\mathsf{x}_{q_a}) = v'(\mathsf{x}_{q_a}) = |C_{i+1}^{-1}(q_a)|$.

Otherwise, $v_{i+1}(\mathsf{x}_{q_a}) = v'(\mathsf{x}_{q_a}) + v'(\mathsf{x}_{(q_a, r_{q_a}(i+1))})$. The only case where $v'(\mathsf{x}_{(q_a,r_{q_a}(i+1))}) \neq v(\mathsf{x}_{(q_a,r_{q_a}(i+1))})$ is if $q' \in Q_W$ and $\mathbf{na}(\rho, i+1, e_q) = (q_a, i+1)$ which is impossible. So $v'(\mathsf{x}_{q_a,r_{q_a}(i+1)}) = v(\mathsf{x}_{q_a,r_{q_a}(i+1)})$ and, since $\lambda \in \mathbf{repr}(C, i)$, $v(\mathsf{x}_{q_a,r_{q_a}(i+1)}) = |E_{q_a,i+1}^{\rho,i}|$. Hence, $v_{i+1}(\mathsf{x}_{q_a}) = v'(\mathsf{x}_{q_a}) + |E_{q_a,i+1}^{\rho,i}| = |C_{i+1}^{-1}(q_a)|$, proving **CondRepr1**.

We now show that **CondRepr3** holds. Let $q_a \in Q_A$. We define an injective function $r'_{q_a} : \mathbf{na\text{-}index\text{-}set}(\rho, i+1, q_a) \to [1, |Q_W|+1]$ as follows.

Recall that

$$\mathbf{na\text{-}index\text{-}set}(\rho, i{+}1, q_a) = \begin{cases} (\mathbf{na\text{-}index\text{-}set}(\rho, i, q_a) \setminus \{i+1\}) \cup & \\ \{\mathbf{na\text{-}index}(\rho, i+1, e_q)\} & \text{if } q' \in Q_W \text{ and } \mathbf{na\text{-}state}(\rho, i+1, e_q) = q_a \\ \mathbf{na\text{-}index\text{-}set}(\rho, i, q_a) \setminus \{i+1\} & \text{otherwise.} \end{cases}$$

The new injective function keeps the labelling of the summaries as in the previous configuration, and if needed, associates the labelling of the newly created summary to the index of the next moment $e_q$ will reach an action state.

$$r'_{q_a}(j) = \begin{cases} r_{q_a}(j) & \text{if } j \in \mathbf{na\text{-}index\text{-}set}(\rho, i, q_a) \setminus \{i+1\} \\ k & \text{otherwise, for the unique } k \text{ such that } (\{q'\}, q_a, k) \in \mathcal{S}' \text{ and } (q_a, k) \notin \mathbf{lab}(\mathcal{S}) \end{cases}$$

- For $j \in \mathbf{na\text{-}index\text{-}set}(\rho, i, q_a) \setminus \{i+1\}$, $v_{i+1}(\mathsf{x}_{q_a,r'_{q_a}(j)}) = v_{i+1}(\mathsf{x}_{q_a,r_{q_a}(j)}) = v'(\mathsf{x}_{q_a,r_{q_a}(j)})$. By definition of $v'$,

$$v'(\mathsf{x}_{q_a,r_{q_a}(j)}) = \begin{cases} v(\mathsf{x}_{q_a,r_{q_a}(j)}) + 1 & \text{if } q' \in Q_W \text{ and } \mathbf{na}(\rho, i+1, e_q) = (q_a, j) \\ v(\mathsf{x}_{q_a,r_{q_a}(j)}) & \text{otherwise.} \end{cases}$$

We know that $v(\mathsf{x}_{q_a,r_{q_a}(j)}) = |E_{q_a,j}^{\rho,i}|$. Moreover,

$$E_{q_a,j}^{\rho,i+1} = \begin{cases} E_{q_a,j}^{\rho,i} \cup \{e_q\} & \text{if } q' \in Q_W \text{ and } \mathbf{na}(\rho, i+1, e_q) = (q_a, j) \\ E_{q_a,j}^{\rho,i} & \text{otherwise.} \end{cases}$$

Hence, $v_{i+1}(\mathsf{x}_{q_a,r'_{q_a}(j)}) = v'(\mathsf{x}_{q_a,r_{q_a}(j)}) = |E_{q_a,j}^{\rho,i+1}|$.

- If $\mathbf{na}(\rho, i+1, e_q) = (q_a, j)$ and $j \notin \mathbf{na\text{-}index\text{-}set}(\rho, i, q_a)$, by definition $v_{i+1}(\mathsf{x}_{q_a,r'_{q_a}(j)}) = v'(\mathsf{x}_{q_a,r'_{q_a}(j)}) = v'(\mathsf{x}_{q_a,k})$. This corresponds to the case 2 of the definition of $v'$, where $v'(\mathsf{x}_{q_a,k}) = v(\mathsf{x}_{q_a,k}) + 1$. By definition of $k$, $k \notin r_p(\mathbf{na\text{-}index\text{-}set}(\rho, i, q_a))$, then, since $\lambda = (\mathcal{S}, v) \in \mathbf{repr}(C, i)$, $v(\mathsf{x}_{q_a,k}) = 0$ and $v_{i+1}(\mathsf{x}_{q_a,r'_{q_a}(j)}) = 1$. Moreover, $E_{q_a,j}^{\rho,i} = \{e \in [1, \#\mathbf{proc}(\rho)] \mid C(e) \in Q_W \text{ and } \mathbf{na}(\rho, i, e) = (q_a, j)\}$; since $j \notin \mathbf{na\text{-}index\text{-}set}(\rho, i, q_a)$, $E_{q_a,j}^{\rho,i} = \emptyset$. Hence, $E_{q_a,j}^{\rho,i+1} = E_{q_a,j}^{\rho,i} \cup \{e_q\} = \{e_q\}$ and $v_{i+1}(\mathsf{x}_{q_a,r'_{q_a}(j)}) = 1 = |E_{q_a,j}^{\rho,i+1}|$.

Let $x \notin r'_{q_a}(\mathbf{na\text{-}index\text{-}set}(\rho, i+1, q_a))$. Then, either $x \notin \mathbf{na\text{-}index\text{-}set}(\rho, i, q_a)$, or $x = r_{q_a}(i+1)$. In the first case, $v(\mathsf{x}_{q_a,x}) = 0$, and $v_{i+1}(\mathsf{x}_{q_a,x}) = v'(\mathsf{x}_{q_a,x}) = v(\mathsf{x}_{q_a,x}) = 0$. In the second case, $v_{i+1}(\mathsf{x}_{q_a,x}) = 0$ by construction.

Finally we show that **CondRepr2** holds, i.e. that $\mathcal{S}' = \{(C_{i+1}(E_{q_a,j}^{\rho,i+1}), q_a, r'_{q_a}(j)) \mid q_a \in Q_A, j \in \mathbf{na\text{-}index\text{-}set}(\rho, i+1, q_a)\}$.

- First, let $S \in \mathcal{S}'$. By construction, either $S \in \mathcal{S}_0$, or there exists $S_0 = (\mathsf{pr}, q_a, r_{q_a}(j)) \in \mathcal{S}_0$ such that $S = (\mathsf{pr} \cup \{q'\}, q_a, r_{q_a}(j))$, or $S = (\{q'\}, q_a, k)$.
  - If $S \in \mathcal{S}_0$ and $\mathbf{na}(\rho, i, q_a, e_q) \neq (q_a, j)$ or $q' \in Q_A$, then $S = {S_0}^{+1}$ for some $S_0 \in \mathcal{S}$. Since $(\mathcal{S}, v) \in \mathbf{repr}(C, i)$, $S_0 = (C(E^{\rho,i}_{q_a,j}), q_a, r_{q_a}(j))$ for some $q_a \in Q_A$ and $j \in \mathbf{na\text{-}index\text{-}set}(\rho, i, q_a)$. Then $S = (C_{i+1}(E^{\rho,i}_{q_a,j}), q_a, r_{q_a}(j))$. Then $E^{\rho,i+1}_{q_a,j} = E^{\rho,i}_{q_a,j}$. Moreover, ${S_0}^{+1} \neq \texttt{Done}$ hence $j \neq i+1$, then $r'_{q_a}(j) = r_{q_a}(j)$. Then $S = (C_{i+1}(E^{\rho,i+1}_{q_a,j}), q_a, r'_{q_a}(j))$.
  - If there exists $S_0 = (\mathsf{pr}, q_a, r_{q_a}(j)) \in \mathcal{S}$ such that $S = (\mathsf{pr} \cup \{q'\}, q_a, r_{q_a}(j))$ and $q' \in Q_W$ and $\mathbf{na}(\rho, i+1, e_q) = (q_a, j)$, then, in that case, $E^{\rho,i+1}_{q_a,j} = E^{\rho,i}_{q_a,j} \cup \{e_q\}$ and $j \neq i+1$ so $r'_{q_a}(j) = r_{q_a}(j)$. Then $S = (C_{i+1}(E^{\rho,i+1}_{q_a,j}), q_a, r'_{q_a}(j))$.
  - Finally, if $S = (\{q'\}, q_a, k)$, and $q' \in Q_W$, with $\mathbf{na}(\rho, i+1, e_q) = (q_a, j)$ and $j \notin \mathbf{na\text{-}index\text{-}set}(\rho, i, q_a)$, then $E^{\rho,i+1}_{q_a,j} = \{e_q\}$ and $r'_{q_a}(j) = k$. Then, $S = (C_{i+1}(E^{\rho,i+1}_{q_a,j}), q_a, r'_{q_a}(j))$.
- Now, let $S = (C_{i+1}(E^{\rho,i+1}_{q_a,j}), q_a, r'_{q_a}(j))$ for some $q_a \in Q_A$, $j \in \mathbf{na\text{-}index\text{-}set}(\rho, i+1, q_a)$.
  - If $j \in \mathbf{na\text{-}index\text{-}set}(\rho, i, q_a)$, then $j \neq i+1$ and $r'_{q_a}(j) = r_{q_a}(j)$. Let $S_0 = (C(E^{\rho,i}_{q_a,j}), q_a, r_{q_a}(j)) \in \mathcal{S}$. Such a summary exists because $\lambda = (\mathcal{S}, v)$ is in $\mathbf{repr}(C, i)$. If $q' \in Q_W$ and $\mathbf{na}(\rho, i+1, e_q) = (q_a, j)$, then, $E^{\rho,i+1}_{q_a,j} = E^{\rho,i}_{q_a,j} \cup \{e_q\}$. Moreover, by construction of $\mathcal{S}'$, $(C_{i+1}(E^{\rho,i}_{q_a,j}) \cup \{q'\}, q_a, r_{q_a}(j)) = (C_{i+1}(E^{\rho,i}_{q_a,j}) \cup C_{i+1}(e_q), q_a, r_{q_a}(j)) = (C_{i+1}(E^{\rho,i+1}_{q_a,j}), q_a, r'_{q_a}(j)) \in \mathcal{S}'$. Otherwise, $E^{\rho,i+1}_{q_a,j} = E^{\rho,i}_{q_a,j}$ and, by construction of $\mathcal{S}'$, ${S_0}^{+1} = (C_{i+1}(E^{\rho,i}_{q_a,j}), q_a, r_{q_a}(j)) = (C_{i+1}(E^{\rho,i+1}_{q_a,j}), q_a, r'_{q_a}(j)) \in \mathcal{S}'$.
  - If $j \notin \mathbf{na\text{-}index\text{-}set}(\rho, i, q_a)$, then $r'_{q_a}(j) = k$ and $(\{q'\}, q_a, k) \in \mathcal{S}'$. Moreover, since $j \in \mathbf{na\text{-}index\text{-}set}(\rho, i+1, q_a) \setminus \mathbf{na\text{-}index\text{-}set}(\rho, i, q_a)$, it means that $q' \in Q_W$ and $\mathbf{na}(\rho, i+1, e_q) = (q_a, j)$. Then $E^{\rho,i+1}_{q_a,j} = \{e_q\}$ and $S = (C_{i+1}(E^{\rho,i+1}_{q_a,j}, q_a, r'_{q_a}(j)) = (\{q'\}, q_a, k) \in \mathcal{S}'$.

**Proof that $\lambda' \rightsquigarrow^* \lambda_{i+1}$.**

The only difference between $\lambda'$ and $\lambda_{i+1}$ is in the valuation. We show that $\lambda' \rightsquigarrow^* \lambda_{i+1}$ by successive transitions from $\Delta_e$ in order to transfer the values of counters associated to summaries that have reached `Done` to the counter associated to the corresponding action state.

Let $\{q_1, \dots, q_n\} \subseteq Q_A$ such that, for all $1 \leq \ell \leq n$, $i+1 \in \mathbf{na\text{-}index\text{-}set}(\rho, i, q_\ell)$. It is the set of action states that are reached by processes in waiting zones in $C_{i+1}$. We build the sequence of valuations $v_0, v_1, \dots, v_n$ with $v_0 = v'$ and $v_n = v_{i+1}$ such that $(\mathcal{S}', v_\ell) \rightsquigarrow^* (\mathcal{S}', v_{\ell+1})$, for all $1 \leq \ell < n$. For all $1 \leq \ell \leq n$, let $k_\ell = r_{q_\ell}(i+1)$. We prove by induction that, for all $1 \leq \ell \leq n$, $v_\ell(\mathsf{x}_{q_{\ell'},k_{\ell'}}) = v_{i+1}(\mathsf{x}_{q_{\ell'},k_{\ell'}})$, and $v_\ell(\mathsf{x}_{q_{\ell'}}) = v_{i+1}(\mathsf{x}_{q_{\ell'}})$ for $1 \leq \ell' \leq \ell$, and for all other counter $v_\ell(\mathsf{x}) = v'(\mathsf{x})$.

Observe that for all $1 \leq \ell \leq n$, $k_\ell \notin r'_{q_\ell}(\mathbf{na\text{-}index\text{-}set}(\rho, i+1, q_\ell))$. Indeed, by construction, if $\mathbf{na}(\rho, i+1, e_q) = (q_\ell, j)$, then $r'_{q_\ell}(j) \neq r_{q_\ell}(i+1)$.

Let $v(\mathsf{x}_{q_1,k_1}) = x$. Then $(\mathcal{S}', v)(\overset{\delta}{\rightsquigarrow})^x(\mathcal{S}', v_1)$ with $\delta(\mathsf{x}_{q_1}) = +1$, $\delta(\mathsf{x}_{q_1,k_1}) = -1$ and $\delta(\mathsf{x}) = 0$ for all other $\mathsf{x}$. Since $k_1 \notin r'_{q_\ell}(\mathbf{na\text{-}index\text{-}set}(\rho, i+1, q_1))$, $(q_1, k_1) \notin \mathbf{lab}(\mathcal{S}')$ (due to the fact that $(\mathcal{S}', v_{i+1})$ is in $\mathbf{repr}(\rho, i+1)$), and $(\mathcal{S}', \delta, \mathcal{S}') \in \Delta_e$. Moreover, $v_1(\mathsf{x}_{q_1}) = v'(\mathsf{x}_{q_1}) + v'(\mathsf{x}_{q_1,k_1}) = v_{i+1}(\mathsf{x}_{q_1})$ and $v_1(\mathsf{x}_{q_1,k_1}) = 0 = v_{i+1}(\mathsf{x}_{q_1,k_1})$.

Let $\ell \geq 1$ and assume that the property is true for $1 \leq \ell' < \ell$. We can show in a similar way that for $\delta(\mathsf{x}_{q_\ell}) = +1$, $\delta(\mathsf{x}_{q_\ell,k_\ell}) = -1$ and $\delta(\mathsf{x}) = 0$ for all other counter $\mathsf{x}$, we have $(\mathcal{S}', \delta, \mathcal{S}') \in \Delta_e$ and $(\mathcal{S}', v_{\ell-1})(\overset{\delta}{\rightsquigarrow})^{x'}(\mathcal{S}', v_\ell)$, with $x' = v(\mathsf{x}_{q_\ell,k_\ell})$. Hence, $v_\ell(\mathsf{x}_{q_\ell}) = v_{\ell-1}(\mathsf{x}_{q_\ell}) + v_{\ell-1}(\mathsf{x}_{q_\ell,k_\ell})$. By induction hypothesis, $v_\ell(\mathsf{x}_{q_\ell}) = v'(\mathsf{x}_{q_\ell}) + v'(\mathsf{x}_{q_\ell,k_\ell}) = v_{i+1}(\mathsf{x}_{q_\ell})$.

Moreover $v_\ell(\mathsf{x}_{q_\ell,k_\ell}) = 0 = v_{i+1}(\mathsf{x}_{q_\ell,k_\ell})$. For all other counter $\mathsf{x}$, $v_\ell(\mathsf{x}) = v_{\ell-1}(\mathsf{x})$, which proves the property.

Thus, we have proved that $(\mathcal{S}', v') \rightsquigarrow^* (\mathcal{S}', v_n)$ with $v_n(\mathsf{x}_{q_\ell,k_\ell}) = v_{i+1}(\mathsf{x}_{q_\ell,k_\ell})$, $v_n(\mathsf{x}_{q_\ell}) = v_{i+1}(\mathsf{x}_{q_\ell})$ for all $1 \leq \ell \leq n$, and $v_n(\mathsf{x}) = v'(\mathsf{x})$ for all other counter $\mathsf{x}$, hence $v_n = v_{i+1}$. ◀

# E Proofs of Section 3.3

The proof of Theorem 3.13 relies on the following lemma.

▶ **Lemma E.1.** *There exist $C_0 \in \mathcal{I}$ and $C_f \in \mathcal{C}$ such that $C_0 \to^* C_f$ in $P_\mathcal{V}$ and $C_f(e) = q_f$ for all $e \in [1, ||C_f||]$ if and only if $(\ell_0, \mathbf{0}) \rightsquigarrow^* (\ell_f, \mathbf{0})$ in $\mathcal{V}$.*

**Proof of Lemma E.1.** Let the run of the VASS be $(\ell_0, v_0) \rightsquigarrow (\ell_1, v_1) \rightsquigarrow \cdots \rightsquigarrow (\ell_n, v_n)$ with $v_0 = v_n = \mathbf{0}$ and $\ell_n = \ell_f$. We denote $K = \max_{0 \leq i \leq n} |v_i|$ where $|v_i| = \sum_{\mathsf{x} \in \mathsf{X}} v_i(\mathsf{x})$.

We build an execution of the protocol starting from the initial configuration $C_0$ with $K+1$ processes. The first part of the execution goes as follows.

$C_0 \xrightarrow{1,(q_{in},!!\$,q_1)} C_1 \xrightarrow{2,(q_{in},!!\$,q_1)} \cdots \xrightarrow{K,(q_{in},!!\$,q_1)} C_K \xrightarrow{K+1,(q_{in},!!\mathsf{start},\ell_0)} C_{K+1}$. Observe that $C_{K+1}(j) = \mathsf{zero}$ for all $1 \leq j \leq K$ and $C_{K+1}(K+1) = \ell_0$.

We rename $C_{K+1}$ to $C'_0$.

For all $0 \leq i \leq n$, let $\mathbb{C}(\ell_i, v_i) = \{C \in \mathcal{C} \mid C(K+1) = \ell_i, C([1,K]) = \{\mathsf{zero}, \mathsf{unit}_\mathsf{x} \mid \mathsf{x} \in \mathsf{X}\}$, and for all $\mathsf{x} \in \mathsf{X}, |C^{-1}(\mathsf{unit}_\mathsf{x})| = v_i(\mathsf{x})\}$ be the set of protocol configurations corresponding to $(\ell_i, v_i)$. We show by induction on $i$ that, for all $0 \leq i \leq n$, for all $C \in \mathbb{C}(\ell_i, v_i)$, $C'_0 \to^* C$.

For $i = 0$, observe that $\mathbb{C}(\ell_0, v_0) = \{C'_0\}$ hence the property trivially holds. Let $0 \leq i < n$ and $(\ell_i, \delta_i, \ell_{i+1}) \in \Delta$ such that $(\ell_i, v_i) \rightsquigarrow (\ell_{i+1}, v_{i+1})$ and $v_{i+1} = v_i + \delta_i$. Let $\mathsf{x}_i$ be the unique counter such that $\delta_i(\mathsf{x}_i) \in \{1, -1\}$. Let now $C \in \mathbb{C}(\ell_{i+1}, v_{i+1})$.

**Case $\delta_i(\mathsf{x}_i) = 1$.** Then $v_{i+1}(\mathsf{x}_i) = v_i(\mathsf{x}_i) + 1 > 0$, and since $C \in \mathbb{C}(\ell_{i+1}, v_{i+1})$, there exists $1 \leq e \leq K$ such that $C(e) = \mathsf{unit}_{\mathsf{x}_i}$. Let then $C_i \in \mathcal{C}$ such that $C_i(K+1) = \ell_i$, $C_i(e) = \mathsf{zero}$ and for all other process $1 \leq j \leq K$, $C_i(j) = C(j)$. By definition of the protocol, $(\ell_i, ?\mathsf{inc}_{\mathsf{x}_i}, \ell_{i+1}) \in T$, hence $C_i \xrightarrow{e,(\mathsf{zero},!!\mathsf{inc}_{\mathsf{x}_i},\mathsf{unit}_{\mathsf{x}_i})} C$. Furthermore, it is easy to see that, since $|C_i^{-1}(\mathsf{unit}_{\mathsf{x}_i})| = |C^{-1}(\mathsf{unit}_{\mathsf{x}_i})| - 1 = v_{i+1}(\mathsf{x}_i) - 1 = v_i(\mathsf{x}_i)$, and for all other counter $\mathsf{x}$, $|C_i^{-1}(\mathsf{unit}_\mathsf{x})| = |C^{-1}(\mathsf{unit}_\mathsf{x})| = v_{i+1}(\mathsf{x}) = v_i(\mathsf{x})$, $C_i \in \mathbb{C}(\ell_i, v_i)$. Hence, by induction hypothesis, $C'_0 \to^* C_i \to C$.

**Case $\delta_i(\mathsf{x}_i) = -1$.** Then $v_{i+1}(\mathsf{x}_i) = v_i(\mathsf{x}_i) - 1 < K$ and there exists a process $e$ such that $C(e) = \mathsf{zero}$. Let $C_i \in \mathcal{C}$ such that $C_i(K+1) = \ell_i$, $C_i(e) = \mathsf{unit}_{\mathsf{x}_i}$ and for all other process $1 \leq j \leq K$, $C_i(j) = C(j)$. By definition of the protocol, $(\ell_i, ?\mathsf{dec}_{\mathsf{x}_i}, \ell_{i+1}) \in T$, hence $C_i \xrightarrow{e,(\mathsf{unit}_{\mathsf{x}_i},!!\mathsf{dec}_{\mathsf{x}_i},\mathsf{zero})} C$. Furthermore, it is easy to see that, since $|C_i^{-1}(\mathsf{unit}_{\mathsf{x}_i})| = |C^{-1}(\mathsf{unit}_{\mathsf{x}_i})| + 1 = v_{i+1}(\mathsf{x}_i) + 1 = v_i(\mathsf{x}_i)$, and for all other counter $\mathsf{x}$, $|C_i^{-1}(\mathsf{unit}_\mathsf{x})| = |C^{-1}(\mathsf{unit}_\mathsf{x})| = v_{i+1}(\mathsf{x}) = v_i(\mathsf{x})$, $C_i \in \mathbb{C}(\ell_i, v_i)$. Hence, by induction hypothesis, $C'_0 \to^* C_i \to C$.

Then, since $\mathbb{C}(\ell_n, v_n) = \{C'_n\}$ with $C'_n(K+1) = \ell_n = \ell_f$, and $C'_n(j) = \mathsf{zero}$ for all $1 \leq j \leq K$, we have that $C'_0 \to^* C'_n$. The execution ends in the following way: $C'_n \xrightarrow{1,(\mathsf{zero},!!\mathsf{end},\mathsf{z\text{-}end})} C'_{n+1} \xrightarrow{2,(\mathsf{zero},!!\mathsf{end},\mathsf{z\text{-}end})} \cdots \xrightarrow{K,(\mathsf{zero},!!\mathsf{end},\mathsf{z\text{-}end})} C'_{n+K} \xrightarrow{K+1,(s'_f,!!\mathsf{verif},q_f)} C'_{n+K+1}$. One can observe that for all process $e$, $C'_{n+K+1}(e) = q_f$ which concludes the proof.

We now prove the other direction, which relies on the following intermediate lemma.

▶ **Lemma E.2.** *Let $C_0 \to^* C_f$ such that $C_0 \in \mathcal{I}$ and for all $e \in [1, ||C_f||]$, $C_f(e) = q_f$. Then, the message $\mathsf{verif}$ is broadcast exactly once.*

**Proof.** We let $C_0 \to C_1 \to \cdots \to C_n = C_f$ be the execution that gathers everyone in $q_f$. The broadcast of $\mathsf{verif}$ can be done only with the transition $(\ell'_f, !!\mathsf{verif}, q_f)$.

First, observe that there must exist an index $0 \leq j \leq n$ such that $C_j \xrightarrow{e_1,(\ell'_f,!!\mathsf{verif},q_f)} C_{j+1}$; otherwise, $q_f$ is unreachable. Let $j_1$ denote the first such index. If there exists an index $j_2 > j_1$ such that $C_{j_2} \xrightarrow{e_2,(\ell'_f,!!\mathsf{verif},q_f)} C_{j_2+1}$, then, by construction of the VASS, $e_2 \neq e_1$ (there is now way for $e_1$ to go back to $\ell'_f$), and $C_{j_2+1}(e_1) = \mathsf{err}$. Then, $C_n(e_1) = \mathsf{err}$ which contradicts the definition of the execution. ◀

We let $C_0 \to C_1 \to \cdots \to C_n = C_f$ be the execution that gathers all processes in $q_f$. From Lemma E.2, there exists a unique index $j$ such that $C_j \xrightarrow{e_1,(\ell'_f,!!\mathsf{verif},q_f)} C_{j+1}$. Hence, there exists a unique process $e_1$ evolving in $P_{\mathcal{V}}$; indeed, any process in $P_{\mathcal{V}}$ needs to broadcast $\mathsf{verif}$ in order to reach $q_f$, and the only way to broadcast $\mathsf{verif}$ is to go through $P_{\mathcal{V}}$. So there exists a unique index $j_1 < j$ such that $C_{j_1} \xrightarrow{e_1,(q_{in},!!\mathsf{start},\ell_0)} C_{j_1+1}$. Hence, all other processes receive the message $\mathsf{start}$ in $C_{j_1}$; otherwise they will never reach $q_f$. As a consequence, for each process $e \neq e_1$, $C_{j_1+1}(e) = \mathsf{zero}$.

We now prove that there exists an index $j_1 < j_2 < j$ such that $C_{j_2}(e_1) = \ell_f$ and for any other process $e \neq e_1$, $C_{j_2}(e) = \mathsf{zero}$. First note that, unless $\ell_0 = \ell_f$, there must be at least one process $e$ such that $C_{j_1+1}(e) = \mathsf{zero}$. Otherwise, no process ever broadcasts a message, and $e$ would not be able to reach $\ell_f$ and then $\ell'_f$.

We claim that $j_2$ is the first index such that $C_{j_2} \xrightarrow{e_2,(\mathsf{zero},!!\mathsf{end},\mathsf{z\text{-}end})} C_{j_2+1}$ for some process $e_2$. This transition must occur during the execution, otherwise, processes other than $e_1$ would be unable to reach $q_f$ during the unique broadcast of $\mathsf{verif}$ in the execution. First, we prove that $C_{j_2}(e_1) = \ell_f$. If it is not the case, it needs to receive some messages in the set $\{\mathsf{inc}_\mathsf{x}, \mathsf{dec}_\mathsf{x} \mid \mathsf{x} \in \mathsf{X}\}$ before broadcasting message $!!\mathsf{verif}$. Observe from Lemma E.2, there is exactly one broadcast of $\mathsf{verif}$, and so the sending of one message in $\{\mathsf{inc}_\mathsf{x}, \mathsf{dec}_\mathsf{x} \mid \mathsf{x} \in \mathsf{X}\}$ is received by $e_2$ which goes to state $\mathsf{err}'$. As a consequence, it never reaches $q_f$ which contradicts definition of $C_f$. Hence, $C_{j_2}(e_1) = \ell_f$. We prove now that for all other process $e \neq e_1$, $C_{j_2}(e) = \mathsf{zero}$. If it is not the case, since $\mathsf{end}$ is not broadcast before, there is one process $e'$ such that $C_{j_2}(e') \in \{\mathsf{unit}_\mathsf{x} \mid \mathsf{x} \in \mathsf{X}\}$. In that case, for the same reason as before, it will need to broadcast a message from $\{\mathsf{dec}_\mathsf{x} \mid \mathsf{x} \in \mathsf{X}\}$ while $e_2$ is in $\mathsf{z\text{-}end}$. This would lead $e_2$ to $\mathsf{err}'$ and contradicts the definition of $C_f$.

Hence, $C_{j_2}(e_1) = \ell_f$ and for all other process $e \neq e_1$, $C_{j_2}(e) = \mathsf{zero}$.

We rename the configurations visited during this part of the execution to $C_{j_1+1} = C'_0 \to C'_1 \to \cdots \to C'_k = C_{j_2}$, with $k = j_2 - j_1 - 1$. First observe that since $C'_0(e_1) = \ell_0$, and $C'_k(e_1) = \ell_f$, the structure of the VASS implies that $C'_i(e_1) \in \mathsf{Loc}$ for all $0 \leq i \leq k$. Similarly, since, for all $e \neq e_1$, $C'_0(e) = C'_k(e) = \mathsf{zero}$, the structure of the VASS implies that $C'_i(e) \in \{\mathsf{zero}, \mathsf{unit}_\mathsf{x} \mid \mathsf{x} \in \mathsf{X}\}$, for all $e \neq e_1$, and for all $0 \leq i \leq k$. We build now, by induction on $0 \leq i \leq k$, an execution $(\ell_0, v_0) \rightsquigarrow (\ell_1, v_1) \rightsquigarrow \ldots \rightsquigarrow (\ell_k, v_k)$ of the VASS such that $\ell_i = C'_i(e_1)$, and $v_i(\mathsf{x}) = |C'^{-1}_i(\mathsf{unit}_\mathsf{x})|$ for all $\mathsf{x} \in \mathsf{X}$.

With $v_0 = \mathbf{0}_\mathsf{X}$, the definition of $C'_0$ allows to conclude immediately for the case $i = 0$. Let now $0 \leq i < k$ and assume the property true. Let $C'_i \xrightarrow{e,t} C'_{i+1}$ the transition taken in the execution of the protocol. Since, by construction of the protocol, $C'_i(e_1) \in Q_W$, we know that $e \neq e_1$, and then $t \in \{(\mathsf{zero}, !!\mathsf{inc}_\mathsf{x}, \mathsf{unit}_\mathsf{x}), (\mathsf{unit}_\mathsf{x}, !!\mathsf{dec}_\mathsf{x}, \mathsf{zero}) \mid \mathsf{x} \in \mathsf{X}\}$. We distinguish two cases:

- $t = (\mathsf{zero}, !!\mathsf{inc}_\mathsf{x}, \mathsf{unit}_\mathsf{x})$ for some $\mathsf{x} \in \mathsf{X}$. In that case, by definition of the protocol, $(\ell_i, \delta, \ell_{i+1}) \in \Delta$ with $\delta(\mathsf{x}) = 1$ and $\delta(\mathsf{x}') = 0$ for all other counters $\mathsf{x}'$. Furthermore,

$|C'^{-1}_{i+1}(\mathsf{unit}_\mathsf{x})| = |C'^{-1}_i(\mathsf{unit}_\mathsf{x})| + 1 = v_i(\mathsf{x}) + 1$ by induction hypothesis, and for all other counters $\mathsf{x}'$, $|C'^{-1}_{i+1}(\mathsf{unit}_{\mathsf{x}'})| = |C'^{-1}_i(\mathsf{unit}_{\mathsf{x}'})| = v_i(\mathsf{x}')$, by induction hypothesis. Hence, $(\ell_i, v_i) \rightsquigarrow\!\!\triangleright (\ell_{i+1}, v_{i+1})$ with $v_{i+1}(\mathsf{x}) = |C'^{-1}_{i+1}(\mathsf{unit}_\mathsf{x})|$ for all $\mathsf{x} \in \mathsf{X}$.

- $t = (\mathsf{unit}_\mathsf{x}, !!\mathsf{dec}_\mathsf{x}, \mathsf{zero})$ for some $\mathsf{x} \in \mathsf{X}$. In that case, by definition of the protocol, $(\ell_i, \delta, \ell_{i+1}) \in \Delta$ with $\delta(\mathsf{x}) = -1$ and $\delta(\mathsf{x}') = 0$ for all other counters $\mathsf{x}'$. Furthermore, $|C'^{-1}_{i+1}(\mathsf{unit}_\mathsf{x})| = |C'^{-1}_i(\mathsf{unit}_\mathsf{x})| - 1 = v_i(\mathsf{x}) - 1 \geq 0$, by induction hypothesis. For all other counters $\mathsf{x}'$, $|C'^{-1}_{i+1}(\mathsf{unit}_{\mathsf{x}'})| = |C'^{-1}_i(\mathsf{unit}_{\mathsf{x}'})| = v_i(\mathsf{x}')$, by induction hypothesis. Hence, $(\ell_i, v_i) \rightsquigarrow\!\!\triangleright (\ell_{i+1}, v_{i+1})$ where $v_{i+1}(\mathsf{x}) = |C'^{-1}_{i+1}(\mathsf{unit}_\mathsf{x})|$ for all $\mathsf{x} \in \mathsf{X}$.

Hence, $(\ell_0, v_0) \rightsquigarrow\!\!\triangleright^* (\ell_k, v_k)$ with $\ell_k = C'_k(e_1) = C_{j_2}(e_1) = \ell_f$ and $v_k(\mathsf{x}) = |C'^{-1}_k(\mathsf{unit}_\mathsf{x})| = 0$ for all counters $\mathsf{x}$, which concludes the proof.

◀

## F Proof of Lemma 4.2

We prove that if there are runs $(s_0, \mathbf{0}) \rightsquigarrow\!\!\triangleright^* (s_f, \mathbf{0})$ and $(s_f, \mathbf{0}) \rightsquigarrow\!\!\triangleright^* (s_0, \mathbf{0})$ in the VASS $\mathcal{V}_P$, there is an execution in $P$ that leads all the processes in $q_f$.

The idea is that from a run of the VASS going from $(s_0, \mathbf{0})$ to $(s_f, \mathbf{0})$, one can build a run from $(s_0, \mathbf{0})$ to $(s_f, \mathbf{0})$ *that never takes the transition* $(s_f, \mathbf{0}, s_0)$. Hence this new run will necessarily be of the form: $(s_0, \mathbf{0}) \rightsquigarrow\!\!\triangleright^* (\emptyset, v_{init}) \rightsquigarrow\!\!\triangleright^* (\emptyset, v_f) \rightsquigarrow\!\!\triangleright (s'_f, v'_f) \rightsquigarrow\!\!\triangleright (s_f, v_f) \rightsquigarrow\!\!\triangleright^* (s_f, \mathbf{0})$ with no visit of $(s_f, \mathbf{0}, s_0)$. Here, $v_{init}(\mathsf{x}) = 0$ for all counters $\mathsf{x} \neq \mathsf{x}_{q_{in}}$ and $v_f(\mathsf{x}) = 0$ for all counters $\mathsf{x} \neq \mathsf{x}_{q_f}$. An easy adaptation of the proof of Lemma 3.9 allows then to build an execution of the protocol from $C_0 \in \mathcal{I}$ to $C_f$ such that for all $e \in [1, ||C_f||]$, $C_f(e) = q_f$.

We now focus on transforming the run of the $\mathcal{V}_P$. From the VASS construction, we can decompose the run into: $(s_0, v_0) \rightsquigarrow\!\!\triangleright^* (s_f, v_1) \rightsquigarrow\!\!\triangleright (s_0, v_1) \rightsquigarrow\!\!\triangleright^* (s_f, v_2) \dots (s_0, v_{n-1}) \rightsquigarrow\!\!\triangleright^* (s_f, v_n)$ with $v_0 = v_n = \mathbf{0}$. Moreover, for all $0 \leq i < n$ there exist vectors $v'_i$, $v''_i$ and $v'''_i$ such that: $(s_0, v_i) \rightsquigarrow\!\!\triangleright^* (s_0, v'_i) \rightsquigarrow\!\!\triangleright (\emptyset, v'_i) \rightsquigarrow\!\!\triangleright^* (\emptyset, v''_i) \rightsquigarrow\!\!\triangleright (s'_f, v'''_i) \rightsquigarrow\!\!\triangleright (s_f, v''_i) \rightsquigarrow\!\!\triangleright^* (s_f, v_{i+1})$, with (i) $v'_i(\mathsf{x}_{q_{in}}) = v_i(\mathsf{x}_{q_{in}}) + k$ for some $k \in \mathbb{N}$ and for all other counters $\mathsf{x}$, $v'_i(\mathsf{x}) = v_i(\mathsf{x})$, and (ii) $v''_i(\mathsf{x}_{q_f}) = v_{i+1}(\mathsf{x}_{q_f}) + j$ for some $j \in \mathbb{N}$ and for all other counters $\mathsf{x}$, $v''_i(\mathsf{x}) = v_{i+1}(\mathsf{x})$.

The key observation is to note that for all $s, s' \in \mathsf{Loc}$, for all counter valuations $v_1, \Delta_1, v_2$ such that $\mathbf{0} \preceq \Delta_1$, with $\preceq$ the component-wise order, if $(s, v_1) \rightsquigarrow\!\!\triangleright^* (s', v_2)$ then $(s, v_1 + \Delta_1) \rightsquigarrow\!\!\triangleright^* (s', v_2 + \Delta_1)$. By reordering, we construct a run where all increments of $\mathsf{x}_{q_{in}}$ occur at the beginning and all decrements of $\mathsf{x}_{q_f}$ occur at the end.

For all $0 \leq i < n$, let $\Delta_i$ be defined by:

- $\Delta_i(\mathsf{x}_{q_{in}}) = \sum_{i<j<n} v'_j(\mathsf{x}_{q_{in}}) - v_j(\mathsf{x}_{q_{in}})$
- $\Delta_i(\mathsf{x}_{q_f}) = \sum_{0\leq j<i} v''_j(\mathsf{x}_{q_f}) - v_{j+1}(\mathsf{x}_{q_f})$
- $\Delta_i(\mathsf{x}) = 0$ for all other $\mathsf{x} \in \mathsf{X}$.

Observe first that $\mathbf{0} \preceq \Delta_i$, and that, for all $0 \leq i < n$, $v''_i + \Delta_i = v'_{i+1} + \Delta_{i+1}$. Then, from the different parts of the run $(\emptyset, v'_i) \rightsquigarrow\!\!\triangleright^* (\emptyset, v''_i)$, we deduce that $(\emptyset, v'_i + \Delta_i) \rightsquigarrow\!\!\triangleright^* (\emptyset, v''_i + \Delta_i)$, and we can hence build an execution $(s_0, \mathbf{0}) \rightsquigarrow\!\!\triangleright^* (s_0, v'_0 + \Delta_0) \rightsquigarrow\!\!\triangleright (\emptyset, v'_0 + \Delta_0) \rightsquigarrow\!\!\triangleright^* (\emptyset, v'_1 + \Delta_1) \dots \rightsquigarrow\!\!\triangleright^* (\emptyset, v''_{n-1} + \Delta_{n-1})$.

Since in the original run $(\emptyset, v'_{n-1}) \rightsquigarrow\!\!\triangleright^* (\emptyset, v''_{n-1}) \rightsquigarrow\!\!\triangleright (s'_f, v'''_{n-1}) \rightsquigarrow\!\!\triangleright (s_f, v''_{n-1}) \rightsquigarrow\!\!\triangleright^* (s_f, v_n)$, with $v_n = \mathbf{0}$, we deduce that $v''_{n-1}(\mathsf{x}_{q_{in}}) = \mathbf{0}$, $v''_{n-1}(\mathsf{x}_{q_f}) > 0$, and $v''_{n-1}(\mathsf{x}) = 0$ for all other $\mathsf{x} \in \mathsf{X}$. Moreover, $\Delta_{n-1}(\mathsf{x}) = 0$ for all $\mathsf{x} \neq \mathsf{x}_{q_f}$, hence can continue the execution $(\emptyset, v''_{n-1} + \Delta_{n-1}) \rightsquigarrow\!\!\triangleright (s'_f, v'''_{n-1} + \Delta_{n-1}) \rightsquigarrow\!\!\triangleright (s_f, v''_{n-1} + \Delta_{n-1}) \rightsquigarrow\!\!\triangleright^* (s_f, \mathbf{0})$.

This run avoids the transition $(s_f, \mathbf{0}, s_0)$.

Observe that if $q_f \in Q_W$, then this transformation is not possible, as run segments between $s_0$ and $s_f$ would be of the form $(s_0, v) \rightsquigarrow\!\!\triangleright^* (\emptyset, v_1) \rightsquigarrow\!\!\triangleright^* (\{S_f\}, v_2) \rightsquigarrow\!\!\triangleright (s'_f, v_3) \rightsquigarrow\!\!\triangleright (s_f, v_3)$.

As a result, we can not shorten a run by gluing a configuration with location $\{S_f\}$ and one with location $\emptyset$.

## G Proofs of Section 5

We prove here the soundness and correctness of the reduction presented in Section 5. Denote $\mathcal{L}(\mathcal{A})$ the language of automaton $\mathcal{A}$, i.e. $\mathcal{L}(\mathcal{A}) = \{w \mid \Delta^*(q^0, w) = q^f\}$ where $q^0$ is its initial state and $q^f$ its final state.

▶ **Lemma G.1.** *If there exists a word $w \in \bigcap_{1\leq i\leq n} \mathcal{L}(\mathcal{A}_i)$, then there exists an execution $\rho \in \Lambda_\omega$ and $e \in [1, \#\mathbf{proc}(\rho)]$ such that for all $i \in \mathbb{N}$, there exists $j > i$ such that $\rho_j \xrightarrow{e,(q_{n+2},!!\text{☺},q_1)} \rho_{j+1}$.*

**Proof.** We denote $w = a_1 \dots a_k$. Let $C_0$ be the initial configuration with $n+2$ processes. We form the following prefix of execution:

$$C_0 \xrightarrow{1,(q_{in},!!1,q_1^0)} C_1 \xrightarrow{2,(q_{in},!!2,q_2^0)} \cdots \xrightarrow{n,(q_{in},!!n,q_n^0)} C_n \xrightarrow{n+1,(q_{in},!!\#,q_1)} C_{n+1}$$

$C_{n+1}$ is such that: $C_{n+1}(i) = q_i^0$ for all $i \in [1,n]$, and $C_{n+1}(n+1) = q_1$ and $C_{n+1}(n+2) = q_{in}$. Next, consider the following sequence of configurations:

$$\begin{aligned} C_{n+1} &\xrightarrow{n+2,(q_{in},!!a_1,q_{in})} C_{n+2} \xrightarrow{n+2,(q_{in},!!a_2,q_{in})} \cdots \xrightarrow{n+2,(q_{in},!!a_k,q_{in})} C_{n+k+1} \\ &\xrightarrow{n+2,(q_{in},!!\$,q_{in})} C_{n+k+2} \xrightarrow{1,(q_1^\$,!!\text{end}_1,q_1^e)} \cdots \xrightarrow{n,(q_n^\$,!!\text{end}_n,q_n^e)} C_{2n+k+2} \\ &\xrightarrow{n+1,(q_{n+2},!!\text{☺},q_1)} C_{2n+k+3} \end{aligned}$$

As $a_1 \dots a_k = w \in \bigcap_{1\leq i\leq n} \mathcal{L}(\mathcal{A}_i)$, it holds that: $C_{n+k+1}(i) = q_i^f$ for all $i \in [1,n]$. Hence, $C_{n+k+2}(i) = q_i^\$$ for all $i \in [1,n]$. Furthermore, $C_{n+k+2}(n+1) = q_2$. We conclude that $C_{2n+k+3} = C_{n+1}$. Hence, we can form an infinite execution $\rho = C_0 C_1 \dots C_{n+1} \dots C_{2n+k+3} \dots C_{3n+k+5} \dots$ such that for all $i \in \mathbb{N}$, $C_{n+1+i(n+k+2)} = C_{n+1}$ and $C_{n+i(n+k+2)} \xrightarrow{n+1,(q_{n+2},!!\text{☺},q_1)} C_{n+1+i(n+k+2)}$, which concludes the proof. ◀

▶ **Lemma G.2.** *If there exists an execution $\rho \in \Lambda_\omega$ and $e \in [1, \#\mathbf{proc}(\rho)]$ such that for all $i \in \mathbb{N}$, there exists $j > i$ such that $\rho_j \xrightarrow{e,(q_{n+2},!!\text{☺},q_1)} \rho_{j+1}$, then there exists $w \in \bigcap_{1\leq i\leq n} \mathcal{L}(\mathcal{A}_i)$.*

**Proof.** Denote $\rho$ such an execution. We start by proving the following lemma.

▶ **Lemma G.3.** *There exists $i_0 \in \mathbb{N}$, from which no new processes reach state ☹, in other words, for all process $e$, if $\rho_{i_0}(e) \neq$ ☹ then $\rho_i(e) \neq$ ☹ for all $i > i_0$.*

**Proof.** We reason by contradiction. Otherwise, for all $i_0 \in \mathbb{N}$, either all processes are on ☹, or there exists an process not in ☹ who reach it later. As ☹ is a sink state and there is a finite number of processes, eventually all processes are on ☹, leading to a deadlock configuration, which contradicts the fact that $\rho \in \Lambda_\omega$. ◀

We now denote $e_0$ a process taking infinitely often transition $(q_{n+2}, !!\text{☺}, q_1)$. We easily see that for all $i \in \mathbb{N}$, $\rho_i(e_0) \neq$ ☹, as otherwise, $\rho_j(e_0) \neq$ ☺ for all $j > i$. Consider now $i_1, i_2, i_3 \in \mathbb{N}$ three indexes such that $i_0 < i_1 < i_2 < i_3$. and $\rho_{i_j} \xrightarrow{e_0,(q_{n+2},!!\text{☺},q_1)} \rho_{i_j+1}$ for $j \in \{1,2,3\}$. By construction of the protocol, there exist $e_1, \dots, e_n$ which respectively broadcast messages

$\mathrm{end}_1, \dots, \mathrm{end}_n$ which are received by $e_0$ between $i_1$ and $i_2$. Observe that from the protocol's construction, once process $e_0$ is on state $q_2$, the only sequence of broadcast messages that does not put $e_0$ (before it reaches $q_{n+2}$) in ☹ is $\mathrm{end}_1 \cdot \mathrm{end}_2 \cdots \mathrm{end}_n$. Meaning that there exists $i_1 < i_4 < i_2$ such that $C_{i_4}(e_0) = q_2$ and:

$$C_{i_4} \xrightarrow{e_1,(q_1^{\$},!!\mathrm{end}_1,q_1^e)} \xrightarrow{e_2,(q_2^{\$},!!\mathrm{end}_2,q_2^e)} \cdots \xrightarrow{e_n,(q_n^{\$},!!\mathrm{end}_1,q_n^e)} C_{i_4+n}$$

as any other behaviour would put $e_0$ in ☹. Observe that for all $i \in [1,n]$, $C_{i_4+n}(e_i) = q_i^e$. Note that $i_4 + n = i_2$ as if $C_{i_4+n} \xrightarrow{(q,!!m,q')} C_{i_4+n+1}$ with $m \neq ☺$, then $C_{i_4+n}(e_n) = ☹$. Hence:

$C_{i_2}(e_i) = q_i^0$ for all $i \in [1,n]$

$C_{i_2}(e_0) = q_1$

Denote $i_5$ the next index from which \$ is broadcast (it exists as, in particular, $e_0$ needs to take the cycle with $(q_2, ?\mathrm{end}_1, q_3)$).

As $i_3 > i_0$, no process $e_i$ reaches ☹, hence $\rho_{i_5}(e_i) = q_i^f$ as otherwise, $\rho_{i_5+1}(e_i) = ☹$ for all $i \in [1,n]$.

Observe that between $i_3 + 1$ and $i_5$, the only broadcasts are of the form $(q_{in}, !!a, q_{in})$ with $a \in \Sigma$ as any other broadcast would put $e_0$ in ☹.

Hence, the sequence of letters $a \in \Sigma$ broadcast between $i_3 + 1$ and $i_5$ is received by all processes $e_1, \dots, e_n$ (as automata are complete), and thanks to the protocol's construction, it forms a word accepted by all automata. ◀