ORIGINAL

# Verifying Asynchronous Interactions via Communicating Session Automata

Julien Lange[1] and Nobuko Yoshida[2]

[1] University of Kent, Canterbury, United Kingdom
[2] Imperial College London, London, United Kingdom

**Abstract.** The relationship between communicating automata and session types is the cornerstone of many diverse theories and tools, including type checking, code generation, and runtime verification. A serious limitation of session types is that, while endpoint programs interact asynchronously, the underlying property which guarantees safety of session types is too synchronous: it requires a one-to-one synchronisation between send and receive actions. This paper proposes a sound procedure to verify properties of communicating session automata (CSA), i.e., communicating automata that correspond to multiparty session types. We introduce a new *asynchronous* compatibility property for CSA, called $k$-multiparty compatibility ($k$-MC), which is a strict superset of the synchronous multiparty compatibility proposed in the literature. It is decomposed into two bounded properties: (*i*) a condition called *$k$-safety* which guarantees that, within the bound, all sent messages can be received and each automaton can make a move; and (*ii*) a condition called *$k$-exhaustivity* which guarantees that all $k$-reachable send actions can be fired within the bound. We show that $k$-exhaustive systems *soundly and completely* characterise systems where each automaton behaves uniformly for any bound greater or equal to $k$. We show that checking $k$-MC is PSPACE-complete, but can be done efficiently over large systems by using partial order reduction techniques. We demonstrate that several examples from the literature are $k$-MC, but not synchronous compatible.

**Keywords:** verification · message passing concurrency · asynchrony · communicating automata · session types

## 1 Introduction

*Models of asynchronous message passing programs.* Asynchronous message passing has become one of the key features of several modern concurrent programming languages. For instance, Go and Rust provide a message passing mechanism through bounded channels, while Scala/Akka and Erlang adopt the actor model where processes communicate via unbounded mailboxes. Ensuring the correctness of programs written in these languages is notoriously hard. Due to the very high (possibly infinite) number of interleavings between asynchronous interactions among parallel processes, verifying properties over all possible computations is infeasible. To overcome this problem, several recent approaches use state machines or process calculi as abstract models of the behaviours of the asynchronous communications in concurrent programs. Starting from the source code of a program, a model is extracted, either manually or automatically, and its properties are verified using, e.g., model checking tools. This model-based approach has been successfully applied to verify, e.g., Cloud Haskell [4], Erlang [28], Go [49, 50, 62], and P [15].

As one of the most prominent abstract models for asynchronous interactions, this paper studies communicating automata [16] which express point-to-point communications through unbounded first-in-first-out channels. Like many other expressive communication models, most properties are generally undecidable for this model [16,31]. To circumvent the problem, many restrictions and variations of communicating automata have been introduced. Notably, it has been shown that some properties are decidable for two-party half-duplex systems [17], for universally and existentially bounded systems [32,33,47], and for communicating automata with lossy [2, 18] and un-ordered [19] channels, see [57] for a survey.

*Communicating automata and session types.* This paper focuses on a class of communicating automata, called *communicating session automata*, which includes automata corresponding to *asynchronous multiparty session types* [40]. Session types originated as a typing discipline for the $\pi$-calculus [39, 78], where a session type dictates the behaviour of a process wrt. communications. Session types and related theories have been applied to the verification and specification of concurrent and distributed systems through their integration in several mainstream programming languages, e.g., Haskell [55, 66], Erlang [60], F♯ [59], Go [49, 50, 62], Java [42, 43, 46, 77], OCaml [67], C [63], Python [22, 58, 61], and Scala [74, 75]. Communicating automata and asynchronous multiparty session types [40] are closely related: the latter can be seen as a syntactical representation of the former [23] where a sending state corresponds to an internal choice ($\oplus$) and a receiving state to an external choice ($\&$). This correspondence between communicating automata and multiparty session types has become the foundation of many tools centred on session types, e.g., for generating communication API from multiparty session (global) types [42, 43, 59, 74], for detecting deadlocks in message-passing programs [62, 79], and for monitoring session-enabled programs [7, 22, 58, 60, 61].

*Asynchronous multiparty session types are too synchronous.* A key ingredient of the above tools based on communicating automata is a set of sound procedures, called *multiparty compatibility* in [8, 24, 51], which guarantee that communicating automata representing session types interact correctly, which in turn is used to identify correct protocols or detect errors in endpoint programs. These procedures ensure two basic requirements of interest for multiparty session type frameworks: (*i*) every message that is sent can be eventually received and (*ii*) each automaton can always eventually make a move. However, all of these procedures suffer from a severe limitation: they require that for each execution of the system, there must be an equivalent synchronous execution. Hereafter, we

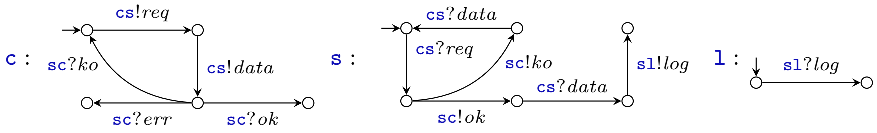


**Fig. 1.** Client-Server-Logger example.

refer to these procedures as *synchronous multiparty compatibility* relations. We explain their limitations with the following example.

*Example 1.* The system in Figure 1 (top) is *not* synchronous multiparty compatible for any of the definitions given in [8, 24, 51]. The figure depicts a system consisting of a client (c), a server (s), and a logger (l), which communicate via unbounded FIFO channels. A transition sr!*a* denotes that a sender puts (asynchronously) a message *a* on channel sr; and a transition sr?*a* denotes the consumption of *a* from channel sr by its receiver. In Figure 1 the client sends a *req*uest to the server, followed by some *data*. It then waits for the server to reply with an *ok* message (in which case the client terminates) or a *ko* message (in which case the client restarts). The server sends a *log* message to the logger only after it has sent an *ok* message to the client. We observe that this system cannot be executed synchronously (i.e., with the restriction that a send action can only be fired when its corresponding receive is ready to be fired). Indeed, for the system to progress further, the client must send some *data* while the server sends either *ok* or *ko*. In fact, due to the asynchronous nature of the communication, this example is rejected by all definitions of synchronous multiparty compatibility as defined in previous works, even though it is safe; hence tools like, e.g., [62, 79] cannot identify the corresponding endpoint programs as safe.

## Contributions

In this work, we focus on communicating automata which are deterministic and whose every state is either sending (internal choice), receiving (external choice), or final. We refer to this class as communicating session automata (CSA), as they cover the most common form of asynchronous multiparty session types [20] (see *Remark 3*), and have been used as a basis to study properties and extensions of session types [8, 9, 24, 42, 43, 53, 54, 58, 60, 61]. Our key discovery is that systems consisting of CSA which preserve the intent of internal and external choices from session types have interesting and tractable properties: in these CSA, whenever an automaton is in a sending state, it can fire any transition, no matter whether channels are bounded; when it is in a receiving state then at most one action must be enabled. For these systems, we can not only introduce a new *asynchronous* multiparty compatibility property which overcomes a fundamental limitation of previous works on session types, but also formally relate session types with several bounded verification approaches for asynchronous programs from the broader area of message passing concurrency [15, 32, 33, 47].

*Asynchronous $k$-multiparty compatibility.* We propose a new definition of multiparty compatibility for CSA, called $k$-multiparty compatibility (or $k$-MC), which generalises synchronous multiparty compatibility definitions, where $k \in \mathbb{N}_{>0}$ is a bound on the number of pending messages in each channel. The definition of $k$-MC relies on $(i)$ *$k$-exhaustivity* which guarantees that all $k$-reachable send actions can be fired within the bound, and $(ii)$ *$k$-safety* which requires that, within the bound $k$, all sent messages can be received and each automaton can always eventually progress. For example, the system in Figure 1 is $k$-multiparty compatible for any $k \in \mathbb{N}_{>0}$, hence it does not lead to communication errors, see Theorem 1. We show that $k$-MC systems include systems that are intrinsically asynchronous and that they enjoy the same safety properties as the ones ensured in the session types literature. We show that, given $k$, deciding $k$-MC is PSPACE-complete (Theorem 2) and that $k$-MC is preserved under partial order reduction (Theorem 6), and thus can be checked effectively. We test several examples from the literature and show that they conform to $k$-MC.

*Relationship with other classes of communicating automata.* The $k$-exhaustivity property plays a central role in enabling us to characterise the relationship between several bounded verification approaches [15, 32, 33, 47]. If a system of CSA validates $k$-exhaustivity, each automaton locally behaves equivalently under any bound greater then or equal to $k$, a property that we call *local bound-agnosticity.* We give a *sound and complete* characterisation of $k$-exhaustivity for CSA in terms of local bound-agnosticity, see Theorem 3. We show that $k$-exhaustive CSA are a strict subset of *existentially* bounded communicating session automata [32,33,47] (an infinite-state sub-class of communicating automata for which some reachability problems are decidable). We show that the two classes coincide for systems in which every message that is sent is eventually received, see Theorem 7. Checking whether a system is $k$-existentially bounded is generally undecidable, even for a given $k$. Therefore, $k$-exhaustivity gives us an effective sufficient condition for existential boundedness. The relationship between $k$-exhaustivity and existential boundedness is used to compare $k$-exhaustivity with $k$-synchronisability, a class of communicating automata recently introduced in [15], which we show to be strictly included in existentially bounded systems, see Theorem 10.

*Synopsis* The rest of the paper is structured as follows. In § 2, we give the necessary background on communicating automata and their properties, and introduce the notions of output/input bound independence which guarantee that internal/external choices are preserved in bounded semantics. In § 3, we introduce the definition of $k$-multiparty compatibility ($k$-MC) and show that $k$-MC systems are safe for systems which validate the bound independence properties. In § 4, we show that $k$-MC can be checked effectively using partial order reduction techniques. In § 5, we relate formally existential boundedness, synchronisability, and $k$-exhaustivity. In § 6 we present an implementation and an experimental evaluation of our theory. We discuss related works in § 7 and conclude in § 8. The appendix contains auxiliary definitions, proofs and additional examples. The implementation of our theory and benchmark data are available online [45].

## 2 Communicating session automata

This section introduces notations and definitions of communicating automata (following [17,51]), as well as the notion of output (resp. input) bound independence which enforces the intent of internal (resp. external) choice in CSA.

Fix a finite set $\mathcal{P}$ of *participants* (ranged over by $\mathtt{p}$, $\mathtt{q}$, $\mathtt{r}$, $\mathtt{s}$, etc.) and a finite alphabet $\Sigma$. The set of *channels* is $\mathcal{C} \overset{\text{def}}{=} \{\mathtt{pq} \mid \mathtt{p}, \mathtt{q} \in \mathcal{P} \text{ and } \mathtt{p} \neq \mathtt{q}\}$, $\mathcal{A} \overset{\text{def}}{=} \mathcal{C} \times \{!, ?\} \times \Sigma$ is the set of *actions* (ranged over by $\ell$), $\Sigma^*$ (resp. $\mathcal{A}^*$) is the set of finite words on $\Sigma$ (resp. $\mathcal{A}$). Let $w$ range over $\Sigma^*$, and $\phi, \psi$ range over $\mathcal{A}^*$. Also, $\epsilon$ ($\notin \Sigma \cup \mathcal{A}$) is the empty word, $|w|$ denotes the length of $w$, and $w \cdot w'$ is the concatenation of $w$ and $w'$ (these notations are overloaded for words in $\mathcal{A}^*$).

**Definition 1 (Communicating automaton).** *A* communicating automaton *is a finite transition system given by a triple $M = (Q, q_0, \delta)$ where $Q$ is a finite set of* states*, $q_0 \in Q$ is the initial state, and $\delta \subseteq Q \times \mathcal{A} \times Q$ is a set of* transitions*.*

The transitions of a communicating automaton are labelled by actions in $\mathcal{A}$ of the form $\mathtt{sr}!a$, representing the *emission* of message $a$ from participant $\mathtt{s}$ to $\mathtt{r}$, or $\mathtt{sr}?a$ representing the *reception* of $a$ by $\mathtt{r}$. Define $\mathit{subj}(\mathtt{pq}!a) = \mathit{subj}(\mathtt{qp}?a) = \mathtt{p}$, $\mathit{obj}(\mathtt{pq}!a) = \mathit{obj}(\mathtt{qp}?a) = \mathtt{q}$, and $\mathit{chan}(\mathtt{pq}!a) = \mathit{chan}(\mathtt{pq}?a) = \mathtt{pq}$. The projection of $\ell$ onto $\mathtt{p}$ is defined as $\pi_{\mathtt{p}}(\ell) = \ell$ if $\mathit{subj}(\ell) = \mathtt{p}$ and $\pi_{\mathtt{p}}(\ell) = \epsilon$ otherwise. Let $\dagger$ range over $\{!, ?\}$, we define: $\pi^{\dagger}_{\mathtt{pq}}(\mathtt{pq}\dagger a) = a$ and $\pi^{\dagger'}_{\mathtt{pq}}(\mathtt{sr}\dagger a) = \epsilon$ if either $\mathtt{pq} \neq \mathtt{sr}$ or $\dagger \neq \dagger'$. We extend these definitions to sequences of actions in the natural way.

A state $q \in Q$ with no outgoing transition is *final*; $q$ is a *sending* (resp. *receiving*) state if it is not final and all its outgoing transitions are labelled with send (resp. receive) actions, and $q$ is a *mixed* state otherwise. Automaton $M = (Q, q_0, \delta)$ is *deterministic* if for all $(q, \ell, q'), (q, \ell', q'') \in \delta : \ell = \ell' \implies q' = q''$. Automaton $M = (Q, q_0, \delta)$ is *send* (resp. *receive*) *directed* if for all sending (resp. receiving) state $q \in Q$ and all $(q, \ell, q'), (q, \ell', q'') \in \delta : \mathit{obj}(\ell) = \mathit{obj}(\ell')$. $M$ is *directed* if it is send and receive directed.

*Remark 1.* In this paper, we consider only deterministic communicating automata without mixed states, and called them *Communicating Session Automata* (CSA). We discuss possible extensions of our results beyond this class in Section 8.

**Definition 2 (System).** *Given a communicating automaton $M_{\mathtt{p}} = (Q_{\mathtt{p}}, q_{0_{\mathtt{p}}}, \delta_{\mathtt{p}})$ for each $\mathtt{p} \in \mathcal{P}$, the tuple $S = (M_{\mathtt{p}})_{\mathtt{p} \in \mathcal{P}}$ is a* system*. A* configuration *of $S$ is a pair $s = (\boldsymbol{q}; \boldsymbol{w})$ where $\boldsymbol{q} = (q_{\mathtt{p}})_{\mathtt{p} \in \mathcal{P}}$ with $q_{\mathtt{p}} \in Q_{\mathtt{p}}$ and where $\boldsymbol{w} = (w_{\mathtt{pq}})_{\mathtt{pq} \in \mathcal{C}}$ with $w_{\mathtt{pq}} \in \Sigma^*$; component $\boldsymbol{q}$ is the* control state *and $q_{\mathtt{p}} \in Q_{\mathtt{p}}$ is the* local state *of automaton $M_{\mathtt{p}}$. The* initial configuration *of $S$ is $s_0 = (\boldsymbol{q_0}; \boldsymbol{\epsilon})$ where $\boldsymbol{q_0} = (q_{0_{\mathtt{p}}})_{\mathtt{p} \in \mathcal{P}}$ and we write $\boldsymbol{\epsilon}$ for the $|\mathcal{C}|$-tuple $(\epsilon, \ldots, \epsilon)$.*

Hereafter, we fix a communicating session automaton $M_{\mathtt{p}} = (Q_{\mathtt{p}}, q_{0_{\mathtt{p}}}, \delta_{\mathtt{p}})$ for each participant $\mathtt{p} \in \mathcal{P}$ and let $S = (M_{\mathtt{p}})_{\mathtt{p} \in \mathcal{P}}$ be the corresponding system. For each $\mathtt{p} \in \mathcal{P}$, we assume that for all $(q, \ell, q') \in \delta_{\mathtt{p}} : \mathit{subj}(\ell) = \mathtt{p}$. Given a configuration $s$ we assume that its components are named consistently, e.g., for $s' = (\boldsymbol{q'}; \boldsymbol{w'})$, we implicitly assume that $\boldsymbol{q'} = (q'_{\mathtt{p}})_{\mathtt{p} \in \mathcal{P}}$ and $\boldsymbol{w'} = (w'_{\mathtt{pq}})_{\mathtt{pq} \in \mathcal{C}}$. We take the convention that $s_0$ denotes the initial configuration of $S$.

**Definition 3 (Reachable configuration).** *A configuration $s' = (\boldsymbol{q}'; \boldsymbol{w}')$ is* reachable *from another configuration $s = (\boldsymbol{q}; \boldsymbol{w})$ by* firing transition *$\ell$, written $s \xrightarrow{\ell} s'$ (or $s \rightarrow s'$ if the label is not relevant), if there are $\mathtt{s}, \mathtt{r} \in \mathcal{P}$ and $a \in \Sigma$ such that either:*

1. *$\ell = \mathtt{sr}!a$ and $(q_{\mathtt{s}}, \ell, {q_{\mathtt{s}}}') \in \delta_{\mathtt{s}}$,*
   - *(a) $q'_{\mathtt{p}} = q_{\mathtt{p}}$ for all $\mathtt{p} \neq \mathtt{s}$, and*
   - *(b) $w'_{\mathtt{sr}} = w_{\mathtt{sr}} \cdot a$ and $w'_{\mathtt{pq}} = w_{\mathtt{pq}}$ for all $\mathtt{pq} \neq \mathtt{sr}$; or*
2. *$\ell = \mathtt{sr}?a$ and $(q_{\mathtt{r}}, \ell, {q_{\mathtt{r}}}') \in \delta_{\mathtt{r}}$,*
   - *(a) $q'_{\mathtt{p}} = q_{\mathtt{p}}$ for all $\mathtt{p} \neq \mathtt{r}$,*
   - *(b) $w_{\mathtt{sr}} = a \cdot w'_{\mathtt{sr}}$, and $w'_{\mathtt{pq}} = w_{\mathtt{pq}}$ for all $\mathtt{pq} \neq \mathtt{sr}$.*

Condition (1b) puts $a$ on channel $\mathtt{sr}$, while (2b) gets $a$ from channel $\mathtt{sr}$.

*Remark 2.* Hereafter, we assume that any bound $k$ is finite and $k \in \mathbb{N}_{>0}$.

A configuration $(\boldsymbol{q}; \boldsymbol{w})$ is $k$-bounded if $\forall \mathtt{pq} \in \mathcal{C} : |w_{\mathtt{pq}}| \leqslant k$. We write $s_1 \xrightarrow{\ell_1 \cdots \ell_m} s_{m+1}$ when $s_1 \xrightarrow{\ell_1} s_2 \cdots s_m \xrightarrow{\ell_m} s_{m+1}$, for some $s_2, \ldots, s_m$ (with $m \geqslant 0$); and say that the *execution* $\ell_1 \cdots \ell_m$ is *$k$-bounded from* $s_1$ if $\forall 1 \leqslant i \leqslant m{+}1 : s_i$ is $k$-bounded. We write $\rightarrow^*$ for the reflexive and transitive closure of $\rightarrow$. Given $\phi \in \mathcal{A}^*$, we write $\mathtt{p} \notin \phi$ iff $\phi = \phi_0 \cdot \ell \cdot \phi_1 \implies \mathit{subj}(\ell) \neq \mathtt{p}$. We write $s \xrightarrow{\phi}_k s'$ if $s'$ is reachable with a $k$-bounded execution $\phi$ from $s$. The set of *reachable configurations of* $S$ is $\mathit{RS}(S) = \{s \mid s_0 \rightarrow^* s\}$. The *$k$-reachability set of* $S$ is the largest subset $\mathit{RS}_k(S)$ of $\mathit{RS}(S)$ within which each configuration $s$ can be reached by a $k$-bounded execution from $s_0$.

The definition of safety below streamlines notions of safety from previous works [8, 17, 24, 51] (guaranteeing the absence of deadlocks, orphan messages, and unspecified receptions).

**Definition 4 ($k$-Safety).** *$S$ is $k$-*safe *if the conditions below hold for all $s = (\boldsymbol{q}; \boldsymbol{w}) \in \mathit{RS}_k(S)$:*

1. *For all $\mathtt{pq} \in \mathcal{C}$, if $w_{\mathtt{pq}} = a \cdot w'$, then $s \rightarrow_k^* \xrightarrow{\mathtt{pq}?a}_k$.*
2. *For all $\mathtt{p} \in \mathcal{P}$, if $q_{\mathtt{p}}$ is a* receiving *state, then $s \rightarrow_k^* \xrightarrow{\mathtt{qp}?a}_k$ for some $\mathtt{q} \in \mathcal{P}$ and $a \in \Sigma$.*

*We say that $S$ is* safe *if it validates the unbounded version of $k$-safety ($\infty$-safe).*

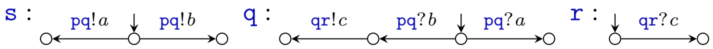
Property (1), called *eventual reception* (ER), requires that any message that is sent can always eventually be received (i.e., if $a$ is the head of a queue then there must be an execution that consumes $a$), and Property (2), called *progress*, requires that any automaton in a receiving state can eventually make a move (i.e., it can always eventually receive an *expected* message).

We say that a configuration $s$ is *stable* iff $s = (\boldsymbol{q}; \boldsymbol{\epsilon})$, i.e., all its queues are empty. Next, we define the *stable property* for systems of communicating automata, following the definition from [24].

**Definition 5 (Stable).** *$S$ has the* stable property *(SP) if $\forall s \in RS(S) : \exists(\boldsymbol{q}; \boldsymbol{\epsilon}) \in RS(S) : s \rightarrow^{*} (\boldsymbol{q}; \boldsymbol{\epsilon})$.*

A system has the stable property if it is possible to reach a stable configuration from any reachable configuration. This property is called *deadlock-free* in [33]. The stable property implies the eventual reception property, but not safety (e.g., an automaton may be waiting for an input in a stable configuration, see Example 2), and safety does not imply the stable property, see Example 4.

*Example 2.* The following system has the stable property, but it is not safe.



Next, we define two properties related to *bound independence*. They specify classes of CSA whose branching behaviours are not affected by channel bounds.

**Definition 6 ($k$-OBI).** *$S$ is $k$-*output bound independent *($k$-OBI), if for all $s = (\boldsymbol{q}; \boldsymbol{w}) \in RS_k(S)$ and $\mathtt{p} \in \mathcal{P}$, if $s \xrightarrow{\mathtt{pq}!a}_k$, then $\forall (q_{\mathtt{p}}, \mathtt{pr}!b, q'_{\mathtt{p}}) \in \delta_{\mathtt{p}} : s \xrightarrow{\mathtt{pr}!b}_k$.*

**Definition 7 ($k$-IBI).** *$S$ is $k$-*input bound independent *($k$-IBI), if for all $s = (\boldsymbol{q}; \boldsymbol{w}) \in RS_k(S)$ and $\mathtt{p} \in \mathcal{P}$, if $s \xrightarrow{\mathtt{qp}?a}_k$, then $\forall \ell \in \mathcal{A} : s \xrightarrow{\ell}_k \wedge subj(\ell) = \mathtt{p} \implies \ell = \mathtt{qp}?a$.*

If $S$ is $k$-OBI, then any automaton that reaches a sending state is able to fire any of its available transitions, i.e., sending states model *internal choices* which are not constrained by bounds greater than or equal to $k$. We note that the unbounded version of $k$-OBI ($k = \infty$) is trivially satisfied for any system due to asynchrony. If $S$ is $k$-IBI, then any automaton that reaches a receiving state is able to fire at most one transition, i.e., receiving states model *external choices* where the behaviour of the receiving automaton is controlled by its environment. We write IBI for the unbounded version of $k$-IBI ($k = \infty$).

Checking the IBI property is generally undecidable. However, systems consisting of (send and receive) *directed* automata are trivially $k$-IBI and $k$-OBI for all $k$, this subclass of CSA was referred to as *basic* in [24]. We introduce larger decidable approximations of IBI in Definitions 11 and 12.

**Proposition 1.** *(1) If $S$ is send directed, then $S$ is $k$-OBI for all $k \in \mathbb{N}_{>0}$. (2) If $S$ is receive directed, then $S$ is IBI (and $k$-IBI for all $k \in \mathbb{N}_{>0}$).*

*Remark 3.* CSA validating $k$-OBI and IBI strictly include the most common forms of asynchronous multiparty session types, e.g., the directed CSA of [24], and systems obtained by projecting Scribble specifications (global types) which need to be receive directed (this is called "consistent external choice subjects" in [43]) and which validate 1-OBI by construction since they are projections of synchronous specifications where choices must be located at a unique sender.

The equivalence relation defined below relates executions which only differ by re-ordering of independent actions, it is used in several results below.

**Definition 8 (Projected equivalence).** *Let $\phi, \psi \in \mathcal{A}^{*}$, we define: $\phi \asymp \psi$* if *$\forall \mathtt{p} \in \mathcal{P} : \pi_{\mathtt{p}}(\phi) = \pi_{\mathtt{p}}(\psi)$.*

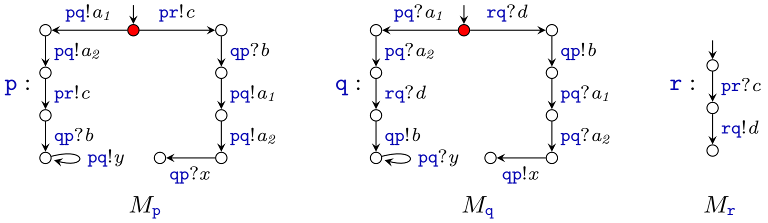


**Fig. 2.** Example of a *non*-IBI and *non*-safe system.

## 3 Bounded compatibility for CSA

In this section, we introduce *$k$-multiparty compatibility* ($k$-MC) and study its properties wrt. safety of communicating session automata (CSA) which are $k$-OBI and IBI. Then, we soundly and completely characterise $k$-exhaustivity in terms of local bound-agnosticity, a property which guarantees that communicating automata behave equivalently under any bound greater than or equal to $k$.

### 3.1 Multiparty compatibility

The definition of $k$-MC is crucially divided in two parts: ($i$) *$k$-exhaustivity* guarantees that the set of $k$-reachable configurations contains enough information for making a sound decision wrt. safety of the system under consideration; and ($ii$) *$k$-safety* (Definition 4) guarantees that a subset of all possible executions is free of any communication errors. Next, we define $k$-exhaustivity, then $k$-multiparty compatibility. Intuitively, a system is $k$-exhaustive if for all $k$-reachable configurations, whenever a send action is enabled, then it can be fired within a $k$-bounded execution.

**Definition 9 ($k$-Exhaustivity).** *$S$ is* $k$-exhaustive *if for all $s = (\boldsymbol{q}; \boldsymbol{w}) \in RS_k(S)$ and $\mathtt{p} \in \mathcal{P}$, if $q_\mathtt{p}$ is a sending state, then $\forall (q_\mathtt{p}, \ell, q'_\mathtt{p}) \in \delta_\mathtt{p} : \exists \phi \in \mathcal{A}^* : s \xrightarrow{\phi}_k \xrightarrow{\ell}_k$ and $\mathtt{p} \notin \phi$.*

**Definition 10 ($k$-Multiparty compatibility).** *$S$ is* $k$-multiparty compatible *($k$-MC) if it is $k$-safe and $k$-exhaustive.*

Definition 10 is a natural extension of the definitions of *synchronous* multiparty compatibility given in [24, Definition 4.2] and [8, Definition 4]. The common key requirements are that *every send* action must be matched by a receive action (i.e., send actions are universally quantified), while *at least one receive* action must find a matching send action (i.e., receive actions are existentially quantified). Here, the universal check on send actions is done via the eventual reception property and the $k$-exhaustivity condition; while the existential check on receive actions is dealt with by the progress property. Checking $k$-exhaustivity is reminiscent of existential boundedness [32,33,47], as it implicitly requires that every execution can be re-ordered in an equivalent $k$-bounded one, see Section 5.

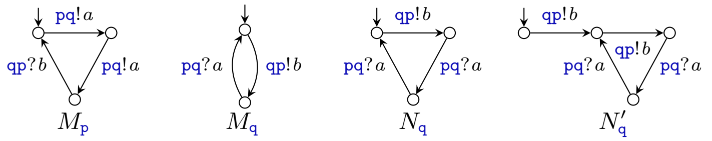


**Fig. 3.** $(M_{\mathtt{p}}, M_{\mathtt{q}})$ is non-exhaustive, $(M_{\mathtt{p}}, N_{\mathtt{q}})$ is 1-exhaustive, $(M_{\mathtt{p}}, N'_{\mathtt{q}})$ is 2-exhaustive.

Whenever systems are $k$-OBI and IBI, then $k$-exhaustivity implies that $k$-bounded executions are sufficient to make a sound decision wrt. safety. This is not necessarily the case for systems outside of this class, see Examples 3 and 5.

*Example 3.* The system $(M_{\mathtt{p}}, M_{\mathtt{q}}, M_{\mathtt{r}})$ in Figure 2 is $k$-OBI for any $k$, but not IBI (it is 1-IBI but not $k$-IBI for any $k \geqslant 2$). When executing with a bound strictly greater than 1, there is a configuration where $M_{\mathtt{q}}$ is in its initial state and *both* its *receive* transitions are enabled. The system is 1-safe and 1-exhaustive (hence 1-MC) but it is *not* 2-exhaustive nor 2-safe. By constraining the automata to execute with a channel bound of 1, the left branch of $M_{\mathtt{p}}$ is prevented to execute together with the right branch of $M_{\mathtt{q}}$. Thus, the fact that the $y$ messages are not received in this case remains invisible in 1-bounded executions. This example can be easily extended so that it is $n$-exhaustive (resp. safe) but not $n{+}1$-exhaustive (resp. safe) by sending/receiving $n{+}1$ $a_i$ messages.

*Example 4.* The system in Figure 1 is *directed* and 1-MC. The system $(M_{\mathtt{p}}, M_{\mathtt{q}})$ in Figure 3 is safe but *not* $k$-MC for any finite $k \in \mathbb{N}_{>0}$. Indeed, for any execution of this system, at least one of the queues grows arbitrarily large. The system $(M_{\mathtt{p}}, N_{\mathtt{q}})$ is 1-MC while the system $(M_{\mathtt{p}}, N'_{\mathtt{q}})$ is *not* 1-MC but it is 2-MC.

*Example 5.* The system in Figure 4 (without the dotted transition) is 1-MC, but not 2-safe; it is not 1-OBI but it is 2-OBI. In 1-bounded executions, $M_{\mathtt{r}}$ can execute $\mathtt{rs}!b \cdot \mathtt{rp}!z$, but it cannot fire $\mathtt{rs}!b \cdot \mathtt{rs}!a$ (queue $\mathtt{rs}$ is full), which violates the 1-OBI property. The system with the dotted transition is not 1-OBI, but it is 2-OBI and $k$-MC for any $k \geqslant 1$. Both systems are receive directed, hence IBI.

Lemma 1 below is key to show that $k$-MC implies safety for $k$-OBI and IBI systems. The proof relies on an intermediate result showing that for any $k{+}1$-reachable configuration $s$, there is a $k$-reachable configuration $t$ (from $s_0$) such that $t$ is $k{+}1$-reachable from $s$. A consequence of this result is that from any reachable configuration of such systems, it is possible to reach a configuration whose queues are bounded by $k$. Hence, these systems are never forced to consume an increasing amount of memory (to store pending messages).

**Lemma 1.** *If $S$ is $k$-OBI, IBI, and $k$-MC, then it is $k{+}1$-OBI and $(k{+}1)$-MC.*

**Theorem 1.** *If $S$ is $k$-OBI, IBI, and $k$-MC, then it is safe.*

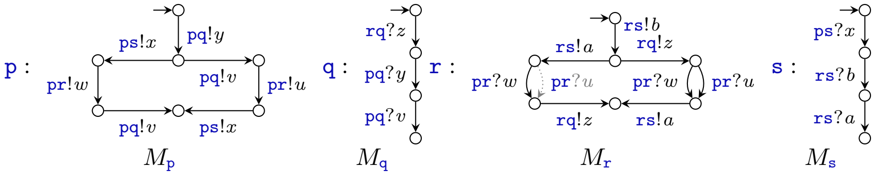


**Fig. 4.** Example of a system which is not 1-OBI.

*Remark 4.* It is undecidable whether there exists a bound $k$ for which an arbitrary system is $k$-MC. This is a consequence of the Turing completeness of communicating (session) automata [16, 31, 54].

Although the IBI property is generally undecidable too, it is possible to identify sound approximations, as we show below. We adapt the dependency relation from [51] and say that action $\ell'$ depends on $\ell$ from $s = (\boldsymbol{q}; \boldsymbol{w})$, written $s \vdash \ell \prec \ell'$, iff $subj(\ell) = subj(\ell') \vee (chan(\ell) = chan(\ell') \wedge w_{chan(\ell)} = \epsilon)$. Action $\ell'$ depends on $\ell$ in $\phi$ from $s$, written $s \vdash \ell \prec_{\phi} \ell'$, if the following holds:

$$s \vdash \ell \prec_{\phi} \ell' \iff \begin{cases} (s \vdash \ell \prec \ell'' \wedge s \vdash \ell'' \prec_{\psi} \ell') \vee s \vdash \ell \prec_{\psi} \ell' & \text{if } \phi = \ell'' \cdot \psi \\ s \vdash \ell \prec \ell' & \text{otherwise} \end{cases}$$

**Definition 11.** *$S$ is $k$-*chained input bound independent *($k$-CIBI) if for all $s = (\boldsymbol{q}; \boldsymbol{w}) \in RS_k(S)$ and $\mathtt{p} \in \mathcal{P}$, if $s \xrightarrow{\mathtt{qp}?a}_k s'$, then $\forall (q_{\mathtt{p}}, \mathtt{sp}?b, q'_{\mathtt{p}}) \in \delta_{\mathtt{p}} : \mathtt{s} \neq \mathtt{q} \implies \neg(s \xrightarrow{\mathtt{sp}?b}_k) \wedge (\forall \phi \in \mathcal{A}^* : s' \xrightarrow{\phi}_k \xrightarrow{\mathtt{sp}!b}_k \implies s \vdash \mathtt{qp}?a \prec_{\phi} \mathtt{sp}!b)$.*

**Definition 12.** *$S$ is $k$-*strong input bound independent *($k$-SIBI) if for all $s = (\boldsymbol{q}; \boldsymbol{w}) \in RS_k(S)$ and $\mathtt{p} \in \mathcal{P}$, if $s \xrightarrow{\mathtt{qp}?a}_k s'$, then $\forall (q_{\mathtt{p}}, \mathtt{sp}?b, q'_{\mathtt{p}}) \in \delta_{\mathtt{p}} : \mathtt{s} \neq \mathtt{q} \implies \neg(s \xrightarrow{\mathtt{sp}?b}_k \vee s' \rightarrow_k^* \xrightarrow{\mathtt{sp}!b}_k)$.*

Definition 11 requires that whenever participant $\mathtt{p}$ can fire a receive action, at most one of its receive actions is enabled at $s$, and no other receive transition from $q_{\mathtt{p}}$ will be enabled until $\mathtt{p}$ has made a move, due to the existence of a dependency chain between the reception of a message and the matching send of another possible reception. Property $k$-SIBI is a slightly stronger version of $k$-CIBI, which may be checked more efficiently. Lemma 2 states that $k$-CIBI (resp. $k$-SIBI), $k$-OBI, and $k$-exhaustivity imply that the IBI property holds. To prove this result, we show that for any system that is $k$-OBI, $k$-CIBI (resp. $k$-SIBI), and $k$-exhaustive, the $k$+1-IBI property holds (by induction on the length of an execution from $s_0$). We show the final result by contradiction, using the key property of $k$-exhaustivity: a $k$-reachable configuration can be reached from any reachable configuration. Figure 5 (right) gives an intuition of the relationships between the different properties.

**Lemma 2.** *If $S$ is $k$-OBI, $k$-CIBI (resp. $k$-SIBI) and $k$-exhaustive, then it is IBI.*

The decidability of the $k$-OBI, $k$-IBI, $k$-SIBI, $k$-CIBI, and $k$-MC conditions is straightforward since both $RS_k(S)$ (which has an exponential number of states wrt. $k$) and $\rightarrow_k$ are finite, given a finite $k$. Theorem 2 states the space complexity of the different procedures, except for $k$-CIBI for which a complexity class is yet to be determined. We show that the properties are PSPACE by reducing to an instance of the reachability problem over a transition system built following the construction of Bollig et al. [11, Theorem 6.3]. The fact that $k$-exhaustivity is PSPACE-hard essentially follows from Theorem 8 and the results by Genest et al. [33, Proposition 5.5]. To show that $k$-OBI, $k$-IBI, $k$-SIBI, and $k$-safety are PSPACE-hard, we reduce the problem of checking if the product of a set of finite state automata has an empty language to checking each property, following a similar construction to the one in [15, Theorem 3].

**Theorem 2.** *The problems of checking the $k$-OBI, $k$-IBI, $k$-SIBI, $k$-safety, and $k$-exhaustivity properties are all decidable and PSPACE-complete (with $k \in \mathbb{N}_{>0}$ given in unary). The problem of checking the $k$-CIBI property is decidable.*

### 3.2 Local bound-agnosticity

We introduce local bound-agnosticity and show that it fully characterises $k$-exhaustive systems. Local bound-agnosticity guarantees that each communicating automaton behave in the same manner for any bound greater than or equal to some $k$. Therefore such systems may executed transparently under a bounded semantics, i.e., the communication model in Go and Rust. First, we define the $k$-bounded transition system of communicating automata and its projection.

**Definition 13 (Transition system).** *The $k$-bounded transition system of $S$ is the labelled transition system $TS_k(S) = (N, s_0, \Delta)$ such that $N = RS_k(S)$, $s_0$ is the initial configuration of $S$, $\Delta \subseteq N \times \mathcal{A} \times N$ is the transition relation, and $(s, \ell, s') \in \Delta$ if and only if $s \xrightarrow{\ell}_k s'$.*

**Definition 14 (Projection).** *Let $\mathcal{T}$ be a labelled transition system (LTS) over $\mathcal{A}$. The* projection *of $\mathcal{T}$ onto $\mathtt{p}$, written $\pi^{\epsilon}_{\mathtt{p}}(\mathcal{T})$, is obtained by replacing each label $\ell$ in $\mathcal{T}$ by $\pi_{\mathtt{p}}(\ell)$.*

Recall that the projection of action $\ell$, written $\pi_{\mathtt{p}}(\ell)$, is defined in Section 2. The automaton $\pi^{\epsilon}_{\mathtt{p}}(TS_k(S))$ is essentially the *local* behaviour of participant $\mathtt{p}$ within the transition system $TS_k(S)$. When each automaton in a system $S$ behaves equivalently for any bound greater than or equal to some $k$, we say that $S$ is *locally bound-agnostic*. Formally, $S$ is locally bound-agnostic for $k$ when $\pi^{\epsilon}_{\mathtt{p}}(TS_k(S))$ and $\pi^{\epsilon}_{\mathtt{p}}(TS_n(S))$ are weakly bisimilar ($\approx$) for each participant $\mathtt{p}$ and any $n \geqslant k$. For $k$-OBI and IBI systems, local bound-agnosticity is a *necessary and sufficient* condition for $k$-exhaustivity, as stated in Theorem 3 and Corollary 1. Corollary 1 is a straightforward consequence of $k$-exhaustivity and Theorem 3.

**Theorem 3.** *Let $S$ be a system.*

*(1) If $\exists k \in \mathbb{N}_{>0} : \forall \mathtt{p} \in \mathcal{P} : \pi^{\epsilon}_{\mathtt{p}}(TS_k(S)) \approx \pi^{\epsilon}_{\mathtt{p}}(TS_{k+1}(S))$, then $S$ is $k$-exhaustive.*

*(2) If $S$ is $k$-OBI, IBI, and $k$-exhaustive, then $\forall \mathtt{p} \in \mathcal{P} : \pi^{\epsilon}_{\mathtt{p}}(TS_k(S)) \approx \pi^{\epsilon}_{\mathtt{p}}(TS_{k+1}(S))$.*

**Corollary 1.** *Let $S$ be $k$-OBI and IBI such that:*
$\exists k \in \mathbb{N}_{>0} : \forall \mathtt{p} \in \mathcal{P} : \pi^{\epsilon}_{\mathtt{p}}(TS_k(S)) \approx \pi^{\epsilon}_{\mathtt{p}}(TS_{k+1}(S))$.
*Then, $\forall n \geqslant k : \forall \mathtt{p} \in \mathcal{P} : \pi^{\epsilon}_{\mathtt{p}}(TS_k(S)) \approx \pi^{\epsilon}_{\mathtt{p}}(TS_n(S))$.*

We note that Theorem 3 (1) is reminiscent of the (PSPACE-complete) verification procedure for existentially bounded systems that have the stable property [33] (an *undecidable* property). However, recall that $k$-exhaustivity is not sufficient to make a sound decision wrt. safety, see Examples 3 and 5. We give an effective procedure to check $k$-exhaustivity and related properties in Section 4.

## 4 Partial order reduction for CSA

In this section, we give a partial order reduction algorithm that allow us to mitigate the exponential cost of checking $k$-MC (wrt. the bound $k$). Partial order reduction is a classical technique to reduce the explored state space in model checking by exploiting the commutativity of independent actions [68].

Next, we define function *partition*$(s)$ which partitions the transitions enabled at $s$, grouping them by subject and arranging them into a sorted list.

**Definition 15 (Partition).** *Let $S$, $s \in RS_k(S)$, and $TS_k(S) = (N, s_0, \Delta)$. The partition of the enabled transitions at $s$ is partition$(s) \overset{def}{=} L_1 \cdots L_n$ such that*

1. $\{\ell \mid s \xrightarrow{\ell}_k s'\} = \bigcup_{1 \leqslant i \leqslant n} L_i$
2. $\forall 1 \leqslant i \neq j \leqslant n :$
   (a) $L_i \cap L_j = \varnothing$ *and*
   (b) $\ell_i \in L_i, \ell_j \in L_j \implies \mathit{subj}(\ell_i) \neq \mathit{subj}(\ell_j)$
3. $\forall 1 \leqslant i \leqslant n : \ell, \ell' \in L_i \implies \mathit{subj}(\ell) = \mathit{subj}(\ell')$
4. $\forall 1 \leqslant i < j \leqslant n : |L_i| \leqslant |L_j|$

In Definition 15, Conditions (1) and (2a) specify that the family of sets $\{L_i\}_{1 \leqslant i \leqslant n}$ is a partition of the transitions enabled at $s$. Conditions (2b) and (3) specify that the function groups transitions executed by the same participant together. Condition (4) guarantees that the list is sorted by increasing order of cardinality, to help reduce the number of redundant branches in Algorithm 1. Definition 15 is used in Algorithm 1 which generates the transition relation $\hat{\Delta}$ of a reduced transition system (the set of states is implicit from $\hat{\Delta}$).

**Definition 16 (Reduced transition system).** *The reduced $k$-bounded transition system of $S$ is a labelled transition system $RTS_k(S) = (\hat{N}, s_0, \hat{\Delta})$ which is a sub-graph of $TS_k(S)$ such that $\hat{\Delta}$ is obtained from Algorithm 1 and $\hat{N}$ is the smallest set such that $s_0 \in \hat{N}$ and $s \in \hat{N} \implies \exists (s_1, \ell, s_2) \in \hat{\Delta} : s \in \{s_1, s_2\}$. We write $s \xrightarrow{\ell}_k s'$ iff $(s, \ell, s') \in \hat{\Delta}$.*

```
1  $visited \leftarrow \varnothing$                    // visited states
2  $accum \leftarrow \varnothing$                      // transitions
3  $stack \leftarrow [\langle s_0, [] \rangle]$        // todo
4  while $stack \neq []$ do
5      $\langle s, E \rangle \leftarrow pop(stack)$
6      if $s \notin visited$ then
7          $visited \leftarrow visited \cup \{s\}$
8          if $E = []$ then
9              $E \leftarrow partition(s)$
10         end
11         foreach $\ell \in head(E)$ do
12             $s' \leftarrow succ(s, \ell)$
13             $push(stack, \langle s', tail(E) \rangle)$
14             $accum \leftarrow accum \cup \{(s, \ell, s'\}$
15         end
16     end
17 end
18 return $accum$
```

**Algorithm 1:** Computing $RTS_k(S)$.

$$f(S,\mathcal{T}) = \begin{cases} (\mathit{snd\text{-}dir}(S) \vee k\text{-}\textsc{obi}(S,\mathcal{T})) \\ \wedge \\ (\mathit{rcv\text{-}dir}(S) \vee k\text{-}\textsc{sibi}(S,\mathcal{T}) \\ \qquad\qquad \vee k\text{-}\textsc{cibi}(S,\mathcal{T})) \\ \wedge \\ (k\text{-}\mathit{exhaustive}(S,\mathcal{T})) \end{cases}$$

```
for $1 \leqslant k \leqslant \textsc{max}$ do
    $\mathcal{T} \leftarrow RTS_k(S)$
    if $f(S,\mathcal{T})$ then
        return $S$ is $k$-safe on $\mathcal{T}$
    end
end
return failed
```

**Algorithm 2:** $k$-MC check.

Algorithm 1 is adapted from the *persistent-set selective search* algorithm from [34, Chapter 4], where instead of computing a persistent state for each explored state, we use a partition of enabled transitions. Each $L_i$ in $partition(s)$ can be seen as a persistent set since no transition outside of $L_i$ can affect the ability of transitions in $L_i$ to fire. Storing all enabled transitions in a list that is progressively consumed guarantees that no transition is forever deferred, hence the cycle proviso [68, Condition C3ii] is satisfied.

Algorithm 1 starts by initialising the required data structures in Lines 1-3, i.e., the set of visited states ($visited$) and the set of accumulated transitions ($accum$) are initialised to the empty set, while the $stack$ contains only the pair $\langle s_0, [] \rangle$ consisting of the initial state of $TS_k(S)$ and the empty list. We overload $[]$ so that it denotes the empty list and the empty stack. The algorithm iterates on the content of $stack$ until it is empty. Each element of the stack is a pair containing a state $s$ and a list of sets of transitions. For each pair $\langle s, E \rangle$, if $E$ is empty, then we compute a new partition (Line 9). Then, we iterate over the first set of transitions in $E$ (we assume $head(E) = \varnothing$ when $E = []$), so to generate the successors of $s$ according to $head(E)$, see Lines 11-14. In Line 12, we write $succ(s, \ell)$ for the (unique) configuration $s'$ such that $s \xrightarrow{\ell}_k s'$. In Line 13, the tail of the list $E$ is pushed on the stack along with the successors $s'$. Finally, the algorithm returns a new set of transitions (Line 18).

We adapt the definitions of $k$-OBI and $k$-SIBI to reduced transition systems, the definition of reduced $k$-CIBI is similar (see Definition 26 in the appendix).

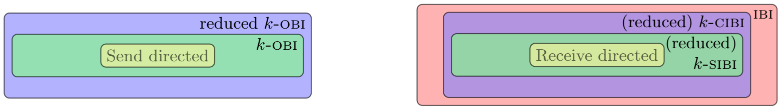


**Fig. 5.** Overview of output and input bounded independence variations.

**Definition 17 (Reduced $k$-OBI).** *Posing $RTS_k(S) = (\hat{N}, s_0, \hat{\Delta})$. System $S$ is* reduced $k$-OBI *if for all $s = (\boldsymbol{q}; \boldsymbol{w}) \in \hat{N}$ and $\mathtt{p} \in \mathcal{P}$, if $s \xrightarrow{\mathtt{pq}!a}_k$, then $\forall (q_{\mathtt{p}}, \mathtt{pr}!b, q'_{\mathtt{p}}) \in \delta_{\mathtt{p}} : s \xrightarrow{\mathtt{pr}!b}_k$.*

**Definition 18 (Reduced $k$-SIBI).** *Posing $RTS_k(S) = (\hat{N}, s_0, \hat{\Delta})$. System $S$ is* reduced $k$-SIBI *if for all $s = (\boldsymbol{q}; \boldsymbol{w}) \in \hat{N}$ and $\mathtt{p} \in \mathcal{P}$, if $s \xrightarrow{\mathtt{qp}?a}_k$, then $\forall (q_{\mathtt{p}}, \mathtt{sp}?b, q'_{\mathtt{p}}) \in \delta_{\mathtt{p}} : \mathtt{s} \neq \mathtt{q} \implies \neg (s \xrightarrow{\mathtt{sp}?b}_k \lor s \rightarrow_k^* \xrightarrow{\mathtt{sp}!b}_k)$.*

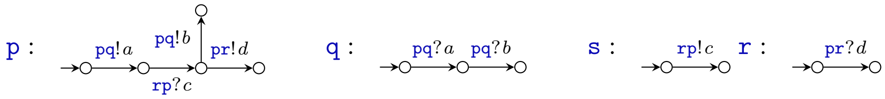
The $k$-SIBI and $k$-CIBI properties (used to approximate IBI) can be decided on the reduced transition system (Theorem 4). The reduced $k$-OBI property is strictly weaker than the $k$-OBI property, see Example 6. However, the *reduced* $k$-OBI property can replace $k$-OBI in Theorem 1 while preserving safety, see Theorem 5. Figure 5 gives an overview of the relationships between the different variations of $k$-OBI, $k$-IBI, and directedness. The inclusions between IBI, $k$-CIBI, and $k$-SIBI hold only for (reduced) $k$-OBI and $k$-exhaustive systems, see Lemma 2.

**Theorem 4.** *Let $S$ be reduced $k$-OBI. $S$ is reduced $k$-CIBI (resp. $k$-SIBI) iff $S$ is $k$-CIBI (resp. $k$-SIBI).*

**Lemma 3.** *Let $S$ be a system, if $S$ is $k$-OBI, then $S$ is also reduced $k$-OBI.*

**Theorem 5.** *If $S$ is reduced $k$-OBI, IBI, and $k$-MC, then it is safe.*

*Example 6.* The system below is reduced 1-OBI, but not 1-OBI. There is a configuration in $TS_1(S)$ from which $M_{\mathtt{p}}$ can fire $\mathtt{pr}!d$ but not $\mathtt{pq}!b$. Depending on the ordering chosen to sort the list of sets of transitions in *partition*(_), $\mathtt{pq}?a$ may always be executed before $M_{\mathtt{p}}$ reaches the violated state in $RTS_1(S)$, hence hiding the violation of $k$-OBI in the reduced transition system.



This system is $k$-exhaustive for any $k \geqslant 1$ and (reduced) $k$-OBI for any $k \geqslant 2$.

Below we adapt the definitions of safety (Definition 4) and $k$-exhaustivity (Definition 9) to reduced transition systems.

**Definition 19 (Reduced $k$-safety).** *Posing $RTS_k(S) = (\hat{N}, s_0, \hat{\Delta})$. System $S$ is* reduced $k$-safe *if the following conditions hold for all $s = (\boldsymbol{q}; \boldsymbol{w}) \in \hat{N}$,*

1. *For all* $\mathtt{pq} \in \mathcal{C}$*, if* $w_{\mathtt{pq}} = a \cdot w'$*, then* $s \rightharpoonup_k{}^* \xrightharpoonup{\mathtt{pq}?a}_k$.
2. *For all* $\mathtt{p} \in \mathcal{P}$*, if* $q_\mathtt{p}$ *is a* receiving *state, then* $s \rightharpoonup_k{}^* \xrightharpoonup{\mathtt{qp}?a}_k$ *for some* $\mathtt{q} \in \mathcal{P}$ *and* $a \in \Sigma$*.*

**Definition 20 (Reduced $k$-exhaustivity).** *Posing* $RTS_k(S) = (\hat{N}, s_0, \hat{\Delta})$*. System* $S$ *is* reduced $k$-exhaustive *if for all* $s = (\boldsymbol{q}; \boldsymbol{w}) \in \hat{N}$ *and* $\mathtt{p} \in \mathcal{P}$*, if* $q_\mathtt{p}$ *is a sending state, then* $\forall (q_\mathtt{p}, \ell, q'_\mathtt{p}) \in \delta_\mathtt{p} : \exists \phi \in \mathcal{A}^* : s \xrightharpoonup{\phi}_k \xrightharpoonup{\ell}_k$ *and* $\mathtt{p} \notin \phi$*.*

Next, we state that checking $k$-safety (resp. $k$-exhaustivity) is equivalent to checking *reduced* $k$-safety (resp. $k$-exhaustivity), which implies that checking $k$-MC can be done on $RTS_k(S)$ instead of $TS_k(S)$, the former being generally much smaller than the latter. We note that the reduction requires (reduced) $k$-OBI and $k$-IBI to hold as they imply that if a transition $(q_\mathtt{p}, \ell, q'_\mathtt{p})$ is enabled at $s = (\boldsymbol{q}; \boldsymbol{w})$, then we have that $(i)$ *all* send actions outgoing from local state $q_\mathtt{p}$ are enabled at $s$ (and they will stay enabled until one is fired) or $(ii)$ *exactly one* receive action is enabled from $q_\mathtt{p}$ (and it will stay enabled until it is fired).

**Theorem 6.** *Let* $S$ *be reduced* $k$-OBI *and reduced* $k$-IBI*. (1)* $S$ *is* reduced $k$*-safe iff* $S$ *is* $k$*-safe. (2)* $S$ *is* reduced $k$*-exhaustive iff* $S$ *is* $k$*-exhaustive.*

Algorithm 2 checks whether a system $S$ is $k$-MC for some $k \leqslant$ MAX, where MAX is a user-provided constant. At each iteration, it constructs the $RTS_k(S)$ of the input system $S$. If $k$ is a sufficient bound to make a sound decision (function $f(S, \mathcal{T})$), then it tests for $k$-safety, otherwise it proceeds to the next iteration with $k{+}1$. Function $f(S, \mathcal{T})$ checks whether the premises of Theorem 5 hold, i.e., if $S$ is not send directed, written *snd-dir*$(S)$, then it checks for $k$-OBI; $S$ is not receive directed, written *rcv-dir*$(S)$, then it checks for $S$-SIBI or $k$-CIBI; then checks whether the $k$-exhaustivity condition holds (all conditions are checked on $RTS_k(S)$).

Finally, we state the optimality of Algorithm 1: it never explores two executions which are $\asymp$-equivalent more than once. Our notion of optimality is slightly different from that of [1] since Algorithm 1 does not use sleep sets.

**Lemma 4.** *Let* $S$ *be a system such that* $RTS_k(S) = (\hat{N}, s_0, \hat{\Delta})$*, for all* $\phi$ *and* $\phi'$ *such that* $s_0 \xrightharpoonup{\phi}_k$ *and* $s_0 \xrightharpoonup{\phi'}_k$*, we have that:* $\phi \asymp \phi' \implies \phi = \phi'$*.*

## 5 Existentially bounded and synchronisable automata

In this section, we formally state the relationships between $k$-exhaustivity, existential boundedness, and synchronisability. Existentially bounded communicating automata [32,33,47] are a class of communicating automata whose executions can always be scheduled in such a way that the number of pending messages is bounded by a given value. The synchronisable systems we study in this section were introduced recently in [15]. Informally, communicating automata are synchronisable if each of their executions can be scheduled in such a way that it consists of sequences of "exchange phases", where each phase consists of a bounded number of send actions, followed by a sequence of receive actions.

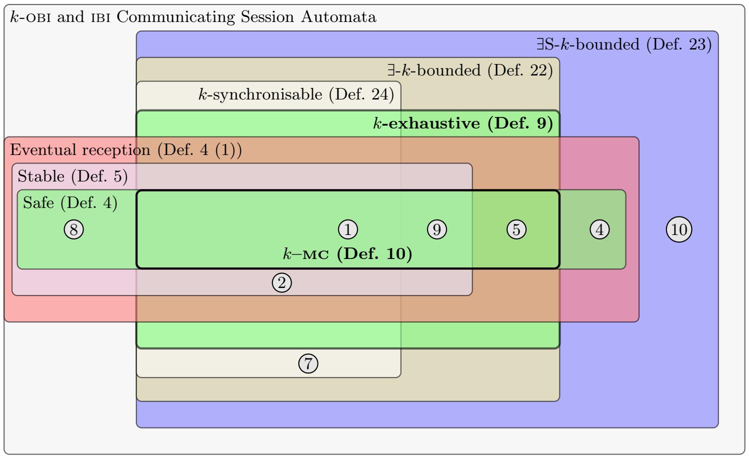


**Fig. 6.** Relationship between $k$-exhaustivity, existentially $k$-boundedness, and $k$-synchronisability in $k$-OBI and IBI CSA (the circled numbers refer to Table 1).

### 5.1 Kuske and Muscholl's existential boundedness

Traditionally, existentially bounded communicating automata are defined on communicating automata that feature (local) accepting states and in terms of *accepting runs*. An accepting run is an execution (starting from $s_0$) which terminates in a configuration $(\boldsymbol{q}; \boldsymbol{w})$ where each $q_{\mathtt{p}}$ is a local accepting state. In our setting, we simply consider that every local state $q_{\mathtt{p}}$ is an accepting state, hence any execution $\phi$ starting from $s_0$ is an accepting run. We first study existential boundedness as defined in [47] as it matches more closely $k$-exhaustivity, we study the "classical" definition of existential boundedness [33] in Section 5.2.

Following [47], we say that an execution $\phi \in \mathcal{A}^*$ is *valid* if for any prefix $\psi$ of $\phi$ and any channel $\mathtt{pq} \in \mathcal{C}$, we have that $\pi^{?}_{\mathtt{pq}}(\psi)$ is a prefix of $\pi^{!}_{\mathtt{pq}}(\psi)$, i.e., an execution is valid if it models the FIFO semantics of communicating automata.

**Definition 21 (Causal equivalence [47]).** *Given $\phi, \psi \in \mathcal{A}^*$, we define: $\phi \doteqdot \psi$* iff *$\phi$ and $\psi$ are* valid *executions and $\phi \asymp \psi$. We write $[\phi]_{\doteqdot}$ for the equivalence class of $\phi$ wrt. $\doteqdot$.*

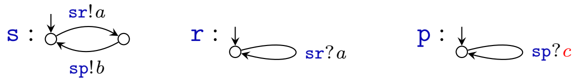
Note that $\doteqdot$ is a congruence on valid executions wrt. concatenation and that any execution starting from $s_0$ is valid.

**Definition 22 (Existentially bounded [47]).** *We say that a valid execution $\phi$ is $k$-*match-bounded *if, for every prefix $\psi$ of $\phi$ the difference between the number of* matched *events of type $\mathtt{pq}!$ and those of type $\mathtt{pq}?$ is bounded by $k$, i.e., $min\{|\pi^{!}_{\mathtt{pq}}(\psi)|, |\pi^{?}_{\mathtt{pq}}(\phi)|\} - |\pi^{?}_{\mathtt{pq}}(\psi)| \leqslant k$.*
*Write $\mathcal{A}^*|_k$ for the set of $k$-match-bounded words. An execution $\phi$ is* existentially *$k$-bounded if $[\phi]_{\doteqdot} \cap \mathcal{A}^*|_k \neq \varnothing$. A system $S$ is existentially $k$-bounded, written $\exists$-$k$-bounded, if each execution in $\{\phi \mid \exists s : s_0 \xrightarrow{\phi} s\}$ is existentially $k$-bounded.*

*Example 7.* Consider Figure 3. $(M_{\mathtt{p}}, M_{\mathtt{q}})$ is *not* existentially $k$-bounded, for any $k$: at least one of the queues must grow infinitely for the system to progress. Systems $(M_{\mathtt{p}}, N_{\mathtt{q}})$ and $(M_{\mathtt{p}}, N'_{\mathtt{q}})$ are existentially bounded since any of their executions can be scheduled to an $\Bumpeq$-equivalent execution which is 2-match-bounded.

Next, we state the relationship between $k$-exhaustivity and existential boundedness which is illustrated in Figure 6 for $k$-OBI and IBI CSA. The circled numbers in the figure refer to key examples summarised in Table 1. Existentially $k$-bounded systems strictly include $k$-exhaustive systems, see the first part of Theorem 7. The strict inclusion is due to systems that do not have the eventual reception property, as we illustrate in Example 8. Recall that the set of $k$-MC systems is strictly included in the set of $k$-exhaustive systems, by Definition 10; hence $k$-MC systems are included in the set of existentially $k$-bounded systems.

*Example 8.* The system below is $\exists$-1-bounded but is not $k$-exhaustive for any $k$.



For any bound $k$, the channel $\mathtt{sp}$ eventually gets full and therefore the send action $\mathtt{sp}!b$ can no longer be fired; hence it does *not* satisfy $k$-exhaustivity. Note that each execution can be reordered into a 1-match-bounded execution since none of the $b$'s are ever matched.

**Theorem 7.** *(1) If $S$ is (reduced) $k$-*OBI*,* IBI*, and $k$-exhaustive, then it is existentially $k$-bounded. (2) If $S$ is existentially $k$-bounded and has the eventual reception property, then it is $k$-exhaustive.*

We show (1) by constructing an existentially bounded execution from an arbitrary execution by using $k$-exhaustivity (to identify an extended $k$-bounded execution) then progressively removing additional actions. For (2), we extend a $k$-bounded execution $\phi$ so that all messages sent in $\phi$ are matched (using the fact that $S$ has the eventual reception property), then use existential boundedness to re-order the extended $\phi$ into a $k$-bounded execution.

### 5.2 Existentially stable bounded communicating automata

The "classical" definition of existentially bounded communicating automata as found in [33] differs slightly from Definition 22, as it relies on a different notion of accepting runs, see [33, page 4]. Assuming that all local states are accepting, we adapt their definition to our setting as follows: a *stable accepting run* is an execution $\phi$ starting from $s_0$ which terminates in a *stable* configuration. We formalise this adaptation in Definition 23.

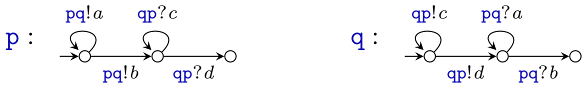
**Definition 23 (Existentially stable bounded [33]).** *A system $S$ is* existentially stable $k$-bounded*, written $\exists S$-$k$-bounded, if for each execution $\phi$ in $\{\phi \mid \exists(\boldsymbol{q};\boldsymbol{\epsilon}) \in RS(S) : s_0 \xrightarrow{\phi} (\boldsymbol{q};\boldsymbol{\epsilon})\}$ there is $\psi$ such that $s_0 \xrightarrow{\psi}_k$ with $\phi \Bumpeq \psi$.*

**Table 1.** Comparison of the properties for key examples ($k$ fixed when required), where direct. stands for directed, OBI for $k$-OBI, SIBI for $k$-SIBI, ER for eventual reception property, SP for stable property, exh. for $k$-exhaustive, ∃(S)-b for ∃ (stable) bounded, and syn. for $n$-synchronisable (for some $n \in \mathbb{N}_{>0}$).

| # | System | Ref. | $k$ | direct. | OBI | SIBI | safe | ER | SP | exh. | ∃S-b | ∃-b | syn. |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | $(M_{\mathtt{c}}, M_{\mathtt{s}}, M_{\mathtt{l}})$ | Fig. 1 | 1 | yes | yes | yes | yes | yes | yes | yes | yes | yes | yes |
| 2 | $(M_{\mathtt{s}}, M_{\mathtt{q}}, M_{\mathtt{r}})$ | Ex. 2 | 1 | yes | yes | yes | no | yes | yes | yes | yes | yes | yes |
| 3 | $(M_{\mathtt{p}}, M_{\mathtt{q}}, M_{\mathtt{r}})$ | Fig. 2 | $\geqslant 3$ | no | yes | no | no | no | no | no | yes | yes | no |
| 4 | $(M_{\mathtt{p}}, M_{\mathtt{q}})$ | Fig. 3 | any | yes | yes | yes | yes | yes | no | no | yes | no | no |
| 5 | $(M_{\mathtt{p}}, N'_{\mathtt{q}})$ | Fig. 3 | 2 | yes | yes | yes | yes | yes | no | yes | yes | yes | no |
| 6 | $(M_{\mathtt{p}}, M_{\mathtt{q}}, M_{\mathtt{r}}, M_{\mathtt{s}})$ | Fig. 4 | 1 | no | yes | yes | yes | yes | no | yes | yes | yes | no |
| 7 | $(M_{\mathtt{s}}, M_{\mathtt{r}}, M_{\mathtt{p}})$ | Ex. 8 | any | yes | yes | yes | no | no | no | no | yes | yes | yes |
| 8 | $(M_{\mathtt{p}}, M_{\mathtt{q}})$ | Ex. 9 | any | yes | yes | yes | yes | yes | yes | no | no | no | no |
| 9 | $(M_{\mathtt{p}}, M_{\mathtt{q}})$ | Ex. 12 | 1 | yes | yes | yes | yes | yes | yes | yes | yes | yes | no |
| 10 | $(M_{\mathtt{p}}, M_{\mathtt{q}}, M_{\mathtt{r}})$ | Fig. 8 | any | yes | yes | yes | no | no | no | no | yes | no | no |

A system is existentially stable $k$-bounded if each of its executions leading to a *stable* configuration can be re-ordered into a $k$-bounded execution (from $s_0$). A key result from [33] is that the problem of testing whether a system is existentially stable $k$-bounded is *undecidable* (whether or not an explicit $k$ is given). However, given a bound $k$ and a system $S$ that has the *stable property*, it is decidable (PSPACE-complete) whether $S$ is existentially stable $k$-bounded. Note that deciding whether a system has the stable property is itself undecidable.

*Example 9.* $(M_{\mathtt{p}}, M_{\mathtt{q}})$ below is not ∃(S)-$k$-bounded, nor $k$-exhaustive, for any $k$.



For instance, execution $\phi$ below is $max\{m, n\}$-bounded. Hence, for any finite $k$, we can generate an execution that is not existentially (stable) $k$-bounded.

$$\phi = \underbrace{\mathtt{pq}!a \cdots \mathtt{pq}!a}_{n \text{ times}} \cdot \mathtt{pq}!b \cdot \underbrace{\mathtt{qp}!c \cdots \mathtt{qp}!c}_{m \text{ times}} \cdot \mathtt{qp}!d \cdot \underbrace{\mathtt{pq}?a \cdots \mathtt{pq}?a}_{n \text{ times}} \cdot \mathtt{pq}?b \cdot \underbrace{\mathtt{qp}?c \cdots \mathtt{qp}?c}_{m \text{ times}} \cdot \mathtt{qp}?d$$

Note that $\phi$ leads to a stable configuration (all sent messages are received).

**Lemma 5.** *Let $S$ be a system and $\phi \in \mathcal{A}^*$ such that $s_0 \xrightarrow{\phi} s = (\boldsymbol{q}; \boldsymbol{\epsilon})$, then $\phi$ is $k$-match-bounded if and only if $\phi$ is $k$-bounded for $s_0$.*

The result below follows from Lemma 5 and the fact that the set of executions that must satisfy boundedness in Definition 23 is included in the set of executions considered in Definition 22.

**Theorem 8.** *(1) If $S$ is existentially $k$-bounded, then it is existentially* stable *$k$-bounded. (2) If $S$ is existentially* stable *$k$-bounded and has the stable property, then it is existentially $k$-bounded.*

We illustrate the relationship between existentially stable bounded communicating automata and the other classes in Figure 6. The examples below further illustrate the strictness of the inclusions, see Table 1 for a summary.

*Example 10.* Consider the systems in Figure 3. $(M_{\mathtt{p}}, M_{\mathtt{q}})$ and $(M_{\mathtt{p}}, N'_{\mathtt{q}})$ are (trivially) existentially stable 1-bounded since none of their (non-empty) executions terminate in a stable configuration. The system $(M_{\mathtt{p}}, N_{\mathtt{q}})$ is existentially stable 2-bounded since each of its executions can be scheduled in such a way that no buffer contains more than 2 messages.

*Example 11.* The system in Example 8 is (trivially) ∃S-1-bounded: none of its (non-empty) executions terminate in a stable configuration ($b$ is never received).

We state the relationship between $k$-exhaustive and existentially stable $k$-bounded systems in Theorem 9 below which relies on Lemma 6.

**Lemma 6.** *Let $S$ be an existentially stable $k$-bounded system with the stable property, then for all $s \in RS_k(S)$, there is $t$ stable such that $s \rightarrow_k{}^{*} t$.*

**Theorem 9.** *Let $S$ be an $\exists$(S)-$k$-bounded system with the stable property, then it is $k$-exhaustive.*

### 5.3 Synchronisable communicating session automata

In this section, we study the relationship between the $k$-synchronisable systems of [15] and $k$-exhaustive systems via existentially bounded communicating automata. The original definition of $k$-synchronisable system [15, Definition 1] is based on communicating automata with *mailbox* semantics, i.e., each automaton has one input queue. Here, we adapt the definition so that it matches our point-to-point semantics. We write $\mathcal{A}_!$ for the set of send actions, i.e., $\mathcal{A} \cap (\mathcal{C} \times \{!\} \times \Sigma)$, and $\mathcal{A}_?$ for the set of receive actions, i.e., $\mathcal{A} \cap (\mathcal{C} \times \{?\} \times \Sigma)$.

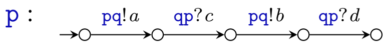
**Definition 24 ($k$-synchronisable).** *A valid execution $\phi = \phi_1 \cdots \phi_n$ is a $k$-*exchange *iff:*

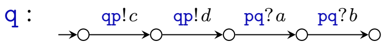
1. $\forall 1 \leqslant i \leqslant n : \phi_i \in \mathcal{A}_!^* \cdot \mathcal{A}_?^* \wedge |\phi_i| \leqslant 2k$
2. $\forall \mathtt{pq} \in \mathcal{C} : \forall 1 \leqslant i \leqslant n : \pi^!_{\mathtt{pq}}(\phi_i) \neq \pi^?_{\mathtt{pq}}(\phi_i) \implies \forall i < j \leqslant n : \pi^?_{\mathtt{pq}}(\phi_j) = \epsilon.$

*We write $\mathcal{A}^*\|_k$ for the set of executions that are $k$-exchanges and say that an execution $\phi$ is $k$-*synchronisable *if $[\phi]_{\Bumpeq} \cap \mathcal{A}^*\|_k \neq \varnothing$. A system $S$ is $k$-*synchronisable *if each execution in $\{\phi \mid \exists s : s_0 \xrightarrow{\phi} s\}$ is $k$-synchronisable.*

Condition (1) says that execution $\phi$ should be a sequence of an arbitrary number of send-receive phases, where each phase consists of at most $2k$ actions. Condition (2) says that if a message is not received in the phase in which it is sent, then it cannot be received in $\phi$. Observe that the bound $k$ is on the number of actions (over possibly different channels) in a phase rather than the number of pending messages in a given channel.

*Example 12.* The system below is 1-MC and $\exists$(S)-1-bounded, but it is *not* $k$-synchronisable for any $k$.





The subsequences of send-receive actions in the $\Bumpeq$-equivalent executions below are highlighted:

$$\begin{aligned}\phi_1 &= \underbrace{\mathtt{pq}!a \cdot \mathtt{qp}!c \cdot \mathtt{qp}?c}_{} \cdot \underbrace{\mathtt{qp}!d \cdot \mathtt{pq}?a}_{} \cdot \underbrace{\mathtt{pq}!b \cdot \mathtt{qp}?d \cdot \mathtt{pq}?b}_{}\\ \phi_2 &= \underbrace{\mathtt{pq}!a \cdot \mathtt{qp}!c \cdot \mathtt{qp}!d \cdot \mathtt{qp}?c \cdot \mathtt{pq}?a}_{} \cdot \underbrace{\mathtt{pq}!b \cdot \mathtt{qp}?d \cdot \mathtt{pq}?b}_{}\end{aligned}$$

Execution $\phi_1$ is 1-bounded for $s_0$, but it is not a $k$-exchange since, e.g., $a$ is received outside of the phase where it is sent (i.e., $\mathtt{pq}!a \cdot \mathtt{qp}!c \cdot \mathtt{qp}?c$). Execution $\phi_2$ is 2-bounded for $s_0$, but it is not a $k$-exchange, because $d$ is received outside of the phase where it is sent. In the terminology of [15], this system is not $k$-synchronisable because there is a receive-send dependency between the exchange of message $c$ and $b$, i.e., $\mathtt{p}$ must receive $c$ before it sends $b$. Hence, there is no execution that is $\Bumpeq$-equivalent to $\phi_1$ and $\phi_2$ and is a $k$-exchange.

We now state the formal relationship between existentially bounded and synchronisable systems, which allows us to relate $k$-exhaustive and synchronisable systems using Theorem 7. Our final result for this section is Theorem 10 which follows easily from Lemma 7 below. The proof of Lemma 7 relies on the facts that $(i)$ the number of send actions is bounded in each send-receive phase and $(ii)$ a message that is un-matched in the phase it is sent can never be received.

**Lemma 7.** *Let $\phi$ be a valid execution. If $\phi$ is a $k$-exchange then it is a $k$-match-bounded execution.*

**Theorem 10.** *(1) If $S$ is $k$-synchronisable, then it is existentially $k$-bounded. (2) If $S$ is $k$-synchronisable and has the eventual reception property, then it is $k$-exhaustive.*

*Example 13.* The (non-IBI) system in Figure 2 is *not* $k$-synchronisable for any $k$, due to executions consisting of the left branch of $M_{\mathtt{p}}$ and the right branch of $M_{\mathtt{q}}$ which are not synchronisable.

*Example 14.* The system $(M_{\mathtt{p}}, M_{\mathtt{q}})$ in Figure 3 is *not* $k$-synchronisable for any $k$. The system $(M_{\mathtt{p}}, N'_{\mathtt{q}})$ is not $k$-synchronisable for any $k$ since the second emission of message $b$ cannot be received in the exchange from which it is sent. Instead, the system $(M_{\mathtt{p}}, N_{\mathtt{q}})$ in Figure 3 is 3-synchronisable since each of its executions can be rescheduled so to consists of the following 3-exchange: $\mathtt{pq}!a \cdot \mathtt{pq}!a \cdot \mathtt{qp}!b \cdot \mathtt{pq}?a \cdot \mathtt{pq}?a \cdot \mathtt{qp}?b$.

Figure 6 and Table 1 summarise the results of § 5 wrt. $k$-OBI and IBI CSA.

**Table 2.** Experimental evaluation. $|\mathcal{P}|$ is the number of participants in the system, $k$ is the bound used for the verification, $|RTS|$ is the number of transitions in $RTS_k(S)$, direct. stands for directed, $k$-OBI stands for *reduced* $k$-OBI, $k$-CIBI stands for *reduced* $k$-CIBI, Time is the time taken to check all the properties shown in this table, and GMC is `yes` if the system is generalised multiparty compatible [51].

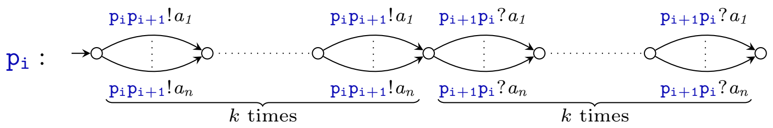
| Example | $\lvert\mathcal{P}\rvert$ | $k$ | $\lvert RTS\rvert$ | *direct.* | $k$-OBI | $k$-CIBI | $k$-MC | Time | GMC |
|---|---|---|---|---|---|---|---|---|---|
| Client-Server-Logger | 3 | 1 | 11 | `yes` | `yes` | `yes` | `yes` | 0.04s | `no` |
| 4 Player game$^\dagger$ [51] | 4 | 1 | 20 | `no` | `yes` | `yes` | `yes` | 0.05s | `yes` |
| Bargain [51] | 3 | 1 | 8 | `yes` | `yes` | `yes` | `yes` | 0.03s | `yes` |
| Filter collaboration [80] | 2 | 1 | 10 | `yes` | `yes` | `yes` | `yes` | 0.03s | `yes` |
| Alternating bit$^\dagger$ [71] | 2 | 1 | 8 | `yes` | `yes` | `yes` | `yes` | 0.04s | `no` |
| TPMContract v2$^\dagger$ [37] | 2 | 1 | 14 | `yes` | `yes` | `yes` | `yes` | 0.04s | `yes` |
| Sanitary agency$^\dagger$ [73] | 4 | 1 | 34 | `yes` | `yes` | `yes` | `yes` | 0.07s | `yes` |
| Logistic$^\dagger$ [65] | 4 | 1 | 26 | `yes` | `yes` | `yes` | `yes` | 0.05s | `yes` |
| Cloud system v4 [36] | 4 | 2 | 16 | `no` | `yes` | `yes` | `yes` | 0.04s | `yes` |
| Commit protocol [15] | 4 | 1 | 12 | `yes` | `yes` | `yes` | `yes` | 0.03s | `yes` |
| Elevator$^\dagger$ [15] | 5 | 1 | 72 | `no` | `yes` | `no` | `yes` | 0.14s | `no` |
| Elevator-dashed$^\dagger$ [15] | 5 | 1 | 80 | `no` | `yes` | `no` | `yes` | 0.16s | `no` |
| Elevator-directed$^\dagger$ [15] | 3 | 1 | 41 | `yes` | `yes` | `yes` | `yes` | 0.07s | `yes` |
| Dev system [70] | 4 | 1 | 20 | `yes` | `yes` | `yes` | `yes` | 0.05s | `no` |
| Fibonacci [59] | 2 | 1 | 6 | `yes` | `yes` | `yes` | `yes` | 0.03s | `yes` |
| SAP-Negot. [59, 64] | 2 | 1 | 18 | `yes` | `yes` | `yes` | `yes` | 0.04s | `yes` |
| SH [59] | 3 | 1 | 30 | `yes` | `yes` | `yes` | `yes` | 0.06s | `yes` |
| Travel agency [59, 76] | 3 | 1 | 21 | `yes` | `yes` | `yes` | `yes` | 0.05s | `yes` |
| HTTP [41, 59] | 2 | 1 | 48 | `yes` | `yes` | `yes` | `yes` | 0.07s | `yes` |
| SMTP [42, 59] | 2 | 1 | 108 | `yes` | `yes` | `yes` | `yes` | 0.08s | `yes` |

## 6 Experimental evaluation

We have implemented our theory in a tool [45] which takes two inputs: $(i)$ a system of communicating automata and $(ii)$ a bound MAX; then applies Algorithm 2 to check whether the CSA are $k$-MC for some $k \leqslant$ MAX.

We have tested our tool on 20 examples taken from the literature, which are reported in Table 2. The table shows that the tool terminates virtually instantaneously on all examples. The table suggests that many systems are indeed $k$-MC and most can be easily adapted to validate bound independence. The examples marked with $^\dagger$ have been slightly modified to make them CSA that validate $k$-OBI and IBI. To remove mixed states, we take only one of the possible interleavings between mixed actions (we take the send action before receive action to preserve safety). The 4 Player game from [51] has been modified so that interleavings of mixed actions are removed (it is the only example of Table 2 that is $k$-CIBI but not $k$-SIBI). The Logistic example from [65, Figure 11.4] has been modified so that the Supplier interacts sequentially (instead of concurrently) with the Shipper then the Consignee. We have added two dummy automata to the Elevator example from [15] which send (resp. receive) messages to (resp. from) the Door so that a mixed state can be removed. The Elevator-dashed example is a variant of the Elevator which is not synchronisable. These examples are not $k$-IBI (for any $k$) because the Elevator automaton can reach a state where it can consume messages sent by different participants (messages `doorClosed` and `openDoor`). This situation cannot occur with a mailbox semantics, as in [15], since each automaton has only one input queue. The Elevator-directed example is another variation where all the automata are directed.

We have assessed the scalability of our approach with automatically generated examples, which we report in Figure 7. Each system considered in these benchmarks consists of $2m$ (directed) CSA for some $m \geqslant 1$ such that $S = (M_{\mathtt{p_i}})_{1 \leqslant \mathtt{i} \leqslant 2m}$, and each automaton $M_{\mathtt{p_i}}$ is of the form (when $i$ is *odd*):


$\mathtt{p_i}$ :
$\mathtt{p_i p_{i+1}}!a_1$
$\mathtt{p_i p_{i+1}}!a_n$
$\mathtt{p_i p_{i+1}}!a_1$
$\mathtt{p_i p_{i+1}}!a_n$
$\mathtt{p_{i+1} p_i}?a_1$
$\mathtt{p_{i+1} p_i}?a_n$
$\mathtt{p_{i+1} p_i}?a_1$
$\mathtt{p_{i+1} p_i}?a_n$
$k$ times
$k$ times

Each $M_{\mathtt{p_i}}$ first sends $k$ messages to participant $\mathtt{p_{i+1}}$, then receives $k$ messages from $\mathtt{p_{i+1}}$. Each message is taken from an alphabet $\{a_1, \dots, a_n\}$ (with $n \geqslant 1$). $M_{\mathtt{p_i}}$ has the same structure when $i$ is *even*, but interacts with $\mathtt{p_{i-1}}$ instead. Observe that any system constructed in this way is $k$-MC for any $k \geqslant 1$, $n \geqslant 1$, and $m \geqslant 1$. The shape of these systems allows us to measure how our approach fares in the worst case (high number of branches and interleavings). Figure 7 gives the time taken for Algorithm 2 to terminate ($y$ axis) wrt. the number of transitions in $RTS_k(S)$ where $k$ is the least natural number for which the system is $k$-MC. Each plot contains a fitted exponential curve which approximates the data points. The plot on the left in Figure 7 gives the timings when $k$ is increasing (every increment from $k = 2$ to $k = 100$) with the other parameters fixed ($n = 1$ and $m = 5$). The middle plot gives the timings when $m$ is increasing (every increment from $m = 1$ to $m = 26$) with the other parameters fixed ($k = 10$ and $n = 1$). The right-hand side plot gives the timings when $n$ is increasing (every increment from $n = 1$ to $n = 10$) with the other parameters fixed ($k = 2$ and $m = 1$). The largest $RTS_k(S)$ on which we have tested our tool has 12222 states and 22220 transitions, and the verification took just under 17 minutes.[3] Observe that partial order reduction mitigates the increasing size of the transition system on which $k$-MC is checked, e.g., these experiments show that parameters $k$ and $m$ have only a linear effect on the number of transitions in $RTS_k(S)$ — see horizontal distances between data points. Unsurprisingly however the number of transitions in $RTS_k(S)$ increases exponentially with $n$.

## 7 Related work

*Theory of communicating automata* Communicating automata were introduced in the 1980s [16] and have since then been studied extensively, namely through their connection with message sequence charts (MSC) [57]. We focus on closely

[3] All the benchmarks in this paper were run on an 8-core Intel i7-7700 machine with 16GB RAM running a 64-bit Linux.

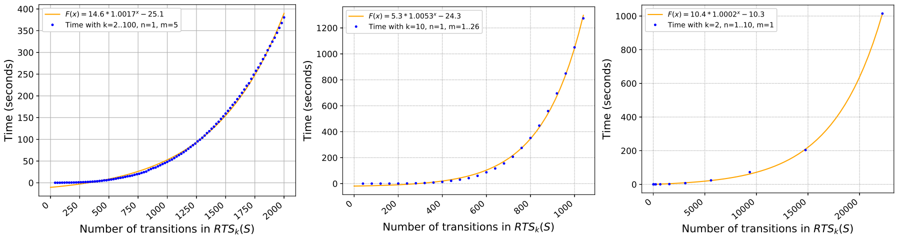
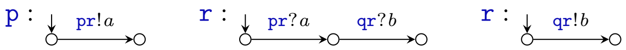


**Fig. 7.** Benchmarks: increasing $k$ (left), increasing $m$ (middle), and increasing $n$ (right).

related works. Several works achieved decidability results by restricting the model. For instance, some of these works substitute reliable and ordered channels with bag or lossy channels [2,3,18,19]. La Torre et al. [48] restrict the topology of the network so that each automaton can only consume messages from one queue (but can send messages to all other queues). Peng and Purushothaman [69] show that reachability, deadlock detection, and un-boundedness detection are decidable for the class of systems where each pair of automata can only exchange one type of message and the topology of the network is a simple cycle. DeYoung and Pfenning [27] investigate a relationship between proofs in a fragment of linear logic and communicating automata that interact via a pipeline topology.

Out of these several variations, existentially bounded communicating automata stand out because they preserve the FIFO semantics of communicating automata, do not restrict the topology of the network, and include systems with an infinite state-space. Existential bounds for MSCs first appeared in [56] and were later applied to the study of communicating automata through MSCs and monadic second order logic in [32,33]. Given a bound $k$ and an arbitrary system of (deterministic) communicating automata $S$, it is generally *undecidable* whether $S$ is existentially $k$-bounded. However, the question becomes decidable when $S$ has the stable property (a property called deadlock-freedom in [33,47]), the problem is PSPACE-complete. The stable property is generally a desirable characteristic, but it is generally *undecidable*. Hence the bounded class is *not* directly applicable to verifying properties of message passing programs since its membership is undecidable overall. We have shown that $(i)$ $k$-OBI, IBI, and $k$-exhaustive CSA systems are (strictly) included in the class of existentially bounded systems, $(ii)$ systems that are existentially bounded (in the sense of [47]) and have the eventual reception property are $k$-exhaustive; and $(iii)$ systems that are existentially stable bounded [33] and have the stable property are $k$-exhaustive. Hence, our work gives a sound *practical* procedure to check whether CSA are existentially bounded. Inspired by the work in [33], Darondeau et al. [21] give decidability results for "data-branching" task systems, which are communicating automata with internal transitions whose only branching states are those where an *internal* choice takes place. The relationship between communicating automata and monadic second order logic was further studied in [10, 12]. To the best of our knowledge, the only tools dedicated to the verification of (unbounded) communicating automata are McScM [38] and Chorgram [52]. Bouajjani et al. [15] study a variation of communicating automata with *mailboxes* (one input queue per automaton). They introduce the class of synchronisable systems and a procedure to check whether a system is $k$-synchronisable; it relies on executions consisting of $k$-bounded exchange phases. Given a system and a bound $k$, it is decidable (PSPACE-complete) whether its executions are equivalent to $k$-synchronous executions. In Section 5.3, we have shown that any $k$-synchronisable system which satisfies eventual reception is also $k$-exhaustive, see Theorem 10. Our characterisation result, based on local bound-agnosticity (Theorem 3), is *unique* to $k$-exhaustivity. It does not apply to existentially boundedness nor synchronisability, see, e.g., Example 8. The term "synchronizability" has been used by Basu et al. [5,6] to refer to another procedure for checking properties of communicating automata with mailboxes. Their notion of synchronizability requires that, for a given system, its synchronous executions are equivalent to its asynchronous executions when considering send actions only. Finkel and Lozes [30] have later shown that this notion of synchronizability is in fact undecidable.

In future work, we would like to study whether our results can be adapted to automata which communicate via mailboxes. We note that a system that is safe with a point-to-point semantics, may not be safe with a mailbox semantics, and vice-versa. For instance, the system in Figure 2 is safe when executed with mailbox semantics. However, the system below is safe in the point-to-point semantics, but *unsafe* with mailbox semantics due to the fact that r may receive $b$ before $a$. To the best of our knowledge, precise relationships and translations between mailbox and point-to-point semantics have yet to be studied.



*Multiparty compatibility* The first definition of multiparty compatibility appeared in [24, Definition 4.2], inspired by the work in [35], to characterise the relationship between global types and communicating automata. This definition was later adapted to the setting of communicating timed automata in [8]. Lange et al. [51] introduced a generalised version of multiparty compatibility (GMC) to support communicating automata that feature mixed or non-directed states. Because our results apply to automata without mixed states, $k$-MC is not a strict extension of GMC, and GMC is not a strict extension of $k$-MC either, as it requires the existence of *synchronous* executions. We discuss how our results may be extended to support communicating automata with mixed states in Section 8. In future work, we will develop an algorithm to synthesise representative choreographies from $k$-MC systems, using the algorithm in [51].

*Communicating automata and programming languages* The notion of multiparty compatibility is at the core of recent works that apply session types techniques to mainstream programming languages. Ng and Yoshida [62] use the multiparty compatibility defined in [51] to detect deadlocks in Go programs. Hu and Yoshida [42] study the well-formedness of Scribble protocols [76] through the

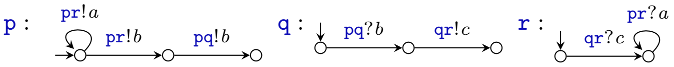


**Fig. 8.** Example of a non ∃-bounded system.

multiparty compatibility of their projections. These protocols are used to generate various endpoint APIs implementing a Scribble specification [42, 43, 59] and to produce runtime monitoring tools [58, 60, 61]. Taylor et al. [79] use multiparty compatibility and choreography synthesis [51] to automate the analysis of the `gen_server` library of Erlang/OTP. We believe that we can transparently widen the set of safe programs captured by these tools by using $k$-MC instead of synchronous multiparty compatibility.

Desai et al. [25] propose a communicating automata-based approach to verify safety properties of programs written in P [26]. Their approach is based on exploring a subset of the (possibly infinite) set of reachable configurations by prioritising certain transitions in order to minimise the size of the queues. Although the approach may not always terminate, they show that it is sound and complete wrt. reachability of error configurations. For instance the system in Figure 8, adapted from [25, Section 9], shows a system for which their approach does *not* terminate. Note that this system is not existentially bounded and therefore it is not $k$-MC for any $k$. It is however trivially existentially stable bounded since no stable configuration is reachable except for the initial one. An interesting area of future work is to investigate similar priority-based executions of CSA systems in order to check the $k$-MC property more efficiently.

D'Osualdo et al. [28] verify safety properties of Erlang programs by inferring a model which abstracts away from message ordering in mailboxes. Their model is based on vector addition systems, for which the reachability problem is decidable. It would be interesting to adapt their approach to infer (mailbox) communicating automata from Erlang programs. Several approaches rely on *sequentialization* of concurrent programs [4, 13, 14, 29, 44, 72], sometimes using bounded executions. For instance, Bouajjani and Emmi [13] verify programs that (asynchronously) send tasks to each other by considering executions bounded by the number of times a sequence of tasks visits the same process. Bakst et al. [4] address the verification of an actor-oriented language (modelled on Erlang and Cloud Haskell) using canonical sequentializations, which over-approximate a program. They show that properties such as deadlock-freedom can be checked efficiently. Their approach requires the program to validate several structural properties, one of which, *symmetric non-determinism*, is reminiscent of receive directedness as it requires every receive action to only receive messages from a single process (or a set of processes running the same code). It would be interesting to relate symmetric non-determinism and directedness more precisely, and consider systems of CSA which consist of several instances of some automaton.

## 8 Conclusions

We have studied communicating session automata via a new condition called $k$-exhaustivity. The $k$-exhaustivity condition is the basis for a new notion of multiparty compatibility, $k$-MC, which captures asynchronous interactions while guaranteeing the two requirements of previous definitions, i.e., for any $k$-MC systems all sent messages can be received and no participant can get permanently stuck. We have shown that $k$-exhaustive systems are fully characterised by local bound-agnosticity, i.e., when each automaton behaves equivalently for any bound greater then or equal to $k$, see Theorem 3. This is relevant for asynchronous message passing programming languages where the possibility of having infinitely many orphan messages is undesirable, in particular in languages such as Go and Rust which require channels to be bounded. We have used the definition of $k$-MC to formally study the relationship between multiparty compatibility for session types with other classes of communicating automata: existentially bounded [33, 47] and synchronisable [15]. We have shown that $k$-MC with $k = 1$ is sufficient to capture several examples from the literature, some of which cannot be verified by previous synchronous multiparty compatibility definitions from [8,24,51]. We have developed a partial order reduction technique to improve the scalability of our approach and demonstrated its performance in an experimental evaluation.

For future work, we plan to support a larger class of communicating automata while preserving our soundness results, Theorem 1 in particular. We believe that it is possible to support mixed states and states which do not satisfy IBI as long as their outgoing transitions are independent (i.e., if they commute). Additionally, to make $k$-MC checking more efficient, we will elaborate heuristics to find optimal bounds and off-load the verification of $k$-MC to an off-the-shelf model checker.

## References

1. P. A. Abdulla, S. Aronis, B. Jonsson, and K. Sagonas. Optimal dynamic partial order reduction. In *POPL 2014*, pages 373–384, 2014.
2. P. A. Abdulla, A. Bouajjani, and B. Jonsson. On-the-fly analysis of systems with unbounded, lossy FIFO channels. In *CAV 1998*, pages 305–318, 1998.
3. P. A. Abdulla and B. Jonsson. Verifying programs with unreliable channels. In *(LICS 1993)*, pages 160–170, 1993.
4. A. Bakst, K. von Gleissenthall, R. G. Kici, and R. Jhala. Verifying distributed programs via canonical sequentialization. *PACMPL*, 1(OOPSLA):110:1–110:27, 2017.
5. S. Basu and T. Bultan. Automated choreography repair. In *FASE 2016*, pages 13–30, 2016.
6. S. Basu, T. Bultan, and M. Ouederni. Deciding choreography realizability. In *POPL 2012*, pages 191–202, 2012.
7. L. Bocchi, T. Chen, R. Demangeon, K. Honda, and N. Yoshida. Monitoring networks through multiparty session types. *Theor. Comput. Sci.*, 669:33–58, 2017.
8. L. Bocchi, J. Lange, and N. Yoshida. Meeting deadlines together. In *CONCUR 2015*, pages 283–296, 2015.

9. L. Bocchi, W. Yang, and N. Yoshida. Timed multiparty session types. In *CONCUR 2014*, pages 419–434, 2014.
10. B. Bollig. Logic for communicating automata with parameterized topology. In *CSL-LICS 2014*, pages 18:1–18:10, 2014.
11. B. Bollig, D. Kuske, and I. Meinecke. Propositional dynamic logic for message-passing systems. *Logical Methods in Computer Science*, 6(3), 2010.
12. B. Bollig and M. Leucker. Message-passing automata are expressively equivalent to EMSO logic. *Theor. Comput. Sci.*, 358(2-3):150–172, 2006.
13. A. Bouajjani and M. Emmi. Bounded phase analysis of message-passing programs. *STTT*, 16(2):127–146, 2014.
14. A. Bouajjani, M. Emmi, and G. Parlato. On sequentializing concurrent programs. In *SAS 2011*, pages 129–145, 2011.
15. A. Bouajjani, C. Enea, K. Ji, and S. Qadeer. On the completeness of verifying message passing programs under bounded asynchrony. In *CAV 2018*, pages 372–391, 2018.
16. D. Brand and P. Zafiropulo. On communicating finite-state machines. *J. ACM*, 30(2):323–342, 1983.
17. G. Cécé and A. Finkel. Verification of programs with half-duplex communication. *Inf. Comput.*, 202(2):166–190, 2005.
18. G. Cécé, A. Finkel, and S. P. Iyer. Unreliable channels are easier to verify than perfect channels. *Inf. Comput.*, 124(1):20–31, 1996.
19. L. Clemente, F. Herbreteau, and G. Sutre. Decidable topologies for communicating automata with FIFO and bag channels. In *CONCUR 2014*, pages 281–296, 2014.
20. M. Coppo, M. Dezani-Ciancaglini, L. Padovani, and N. Yoshida. A Gentle Introduction to Multiparty Asynchronous Session Types. In *15th International School on Formal Methods for the Design of Computer, Communication and Software Systems: Multicore Programming*, volume 9104 of *LNCS*, pages 146–178. Springer, 2015.
21. P. Darondeau, B. Genest, P. S. Thiagarajan, and S. Yang. Quasi-static scheduling of communicating tasks. *Inf. Comput.*, 208(10):1154–1168, 2010.
22. R. Demangeon, K. Honda, R. Hu, R. Neykova, and N. Yoshida. Practical interruptible conversations: distributed dynamic verification with multiparty session types and Python. *Formal Methods in System Design*, 46(3):197–225, 2015.
23. P. Deniélou and N. Yoshida. Multiparty session types meet communicating automata. In *ESOP 2012*, pages 194–213, 2012.
24. P. Deniélou and N. Yoshida. Multiparty compatibility in communicating automata: Characterisation and synthesis of global session types. In *ICALP 2013*, pages 174–186, 2013.
25. A. Desai, P. Garg, and P. Madhusudan. Natural proofs for asynchronous programs using almost-synchronous reductions. In *OOPSLA 2014*, pages 709–725, 2014.
26. A. Desai, V. Gupta, E. K. Jackson, S. Qadeer, S. K. Rajamani, and D. Zufferey. P: safe asynchronous event-driven programming. In *PLDI 2013*, pages 321–332, 2013.
27. H. DeYoung and F. Pfenning. Substructural proofs as automata. In *APLAS 2016*, pages 3–22, 2016.
28. E. D'Osualdo, J. Kochems, and C. L. Ong. Automatic verification of erlang-style concurrency. In *SAS 2013*, pages 454–476, 2013.
29. M. Emmi, A. Lal, and S. Qadeer. Asynchronous programs with prioritized task-buffers. In *SIGSOFT/FSE 2012*, page 48, 2012.
30. A. Finkel and É. Lozes. Synchronizability of communicating finite state machines is not decidable. In *ICALP 2017*, pages 122:1–122:14, 2017.

31. A. Finkel and P. McKenzie. Verifying identical communicating processes is undecidable. *Theor. Comput. Sci.*, 174(1-2):217–230, 1997.
32. B. Genest, D. Kuske, and A. Muscholl. A Kleene theorem and model checking algorithms for existentially bounded communicating automata. *Inf. Comput.*, 204(6):920–956, 2006.
33. B. Genest, D. Kuske, and A. Muscholl. On communicating automata with bounded channels. *Fundam. Inform.*, 80(1-3):147–167, 2007.
34. P. Godefroid. *Partial-Order Methods for the Verification of Concurrent Systems - An Approach to the State-Explosion Problem*, volume 1032 of *Lecture Notes in Computer Science*. Springer, 1996.
35. M. G. Gouda, E. G. Manning, and Y. Yu. On the progress of communications between two finite state machines. *Information and Control*, 63(3):200–216, 1984.
36. M. Güdemann, G. Salaün, and M. Ouederni. Counterexample guided synthesis of monitors for realizability enforcement. In *ATVA 2012*, pages 238–253, 2012.
37. S. Hallé and T. Bultan. Realizability analysis for message-based interactions using shared-state projections. In *SIGSOFT 2010*, pages 27–36, 2010.
38. A. Heußner, T. L. Gall, and G. Sutre. McScM: A general framework for the verification of communicating machines. In *TACAS 2012*, pages 478–484, 2012.
39. K. Honda, V. T. Vasconcelos, and M. Kubo. Language primitives and type discipline for structured communication-based programming. In *ESOP 1998*, pages 122–138, 1998.
40. K. Honda, N. Yoshida, and M. Carbone. Multiparty asynchronous session types. In *POPL 2008*, pages 273–284, 2008.
41. R. Hu. Distributed programming using java apis generated from session types. In *Behavioural Types: from Theory to Tools*. River Publishers, June 2017.
42. R. Hu and N. Yoshida. Hybrid session verification through endpoint API generation. In *FASE 2016*, pages 401–418, 2016.
43. R. Hu and N. Yoshida. Explicit connection actions in multiparty session types. In *FASE 2017*, pages 116–133, 2017.
44. O. Inverso, E. Tomasco, B. Fischer, S. La Torre, and G. Parlato. Bounded model checking of multi-threaded C programs via lazy sequentialization. In *CAV 2014*, pages 585–602, 2014.
45. KMC tool, 2018. https://bitbucket.org/julien-lange/k-checking.
46. D. Kouzapas, O. Dardha, R. Perera, and S. J. Gay. Typechecking protocols with Mungo and StMungo. In *PPDP 2016*, pages 146–159, 2016.
47. D. Kuske and A. Muscholl. Communicating automata. Available at http://eiche.theoinf.tu-ilmenau.de/kuske/Submitted/cfm-final.pdf, 2014.
48. S. La Torre, P. Madhusudan, and G. Parlato. Context-bounded analysis of concurrent queue systems. In *TACAS 2008*, pages 299–314, 2008.
49. J. Lange, N. Ng, B. Toninho, and N. Yoshida. Fencing off Go: liveness and safety for channel-based programming. In *POPL 2017*, pages 748–761, 2017.
50. J. Lange, N. Ng, B. Toninho, and N. Yoshida. A static verification framework for message passing in Go using behavioural types. In *ICSE 2018*. ACM, 2018.
51. J. Lange, E. Tuosto, and N. Yoshida. From communicating machines to graphical choreographies. In *POPL 2015*, pages 221–232, 2015.
52. J. Lange, E. Tuosto, and N. Yoshida. A tool for choreography-based analysis of message-passing software. In *Behavioural Types: from Theory to Tools*. River Publishers, June 2017.
53. J. Lange and N. Yoshida. Characteristic formulae for session types. In *TACAS 2016*, pages 833–850, 2016.

54. J. Lange and N. Yoshida. On the undecidability of asynchronous session subtyping. In *FOSSACS 2017*, pages 441–457, 2017.
55. S. Lindley and J. G. Morris. Embedding session types in Haskell. In *Haskell 2016*, pages 133–145, 2016.
56. M. Lohrey and A. Muscholl. Bounded MSC communication. *Inf. Comput.*, 189(2):160–181, 2004.
57. A. Muscholl. Analysis of communicating automata. In *LATA 2010*, pages 50–57, 2010.
58. R. Neykova, L. Bocchi, and N. Yoshida. Timed Runtime Monitoring for Multiparty Conversations. *FAOC*, pages 1–34, 2017.
59. R. Neykova, R. Hu, N. Yoshida, and F. Abdeljallal. A Session Type Provider: Compile-time API Generation for Distributed Protocols with Interaction Refinements in F♯. In *CC 2018*. ACM, 2018.
60. R. Neykova and N. Yoshida. Let It Recover: Multiparty Protocol-Induced Recovery. In *CC 2017*, pages 98–108. ACM, 2017.
61. R. Neykova and N. Yoshida. Multiparty Session Actors. *LMCS*, 13:1–30, 2017.
62. N. Ng and N. Yoshida. Static deadlock detection for concurrent go by global session graph synthesis. In *CC 2016*, pages 174–184, 2016.
63. N. Ng, N. Yoshida, and K. Honda. Multiparty session C: safe parallel programming with message optimisation. In *TOOLS 2012*, pages 202–218, 2012.
64. Ocean Observatories Initiative. www.oceanobservatories.org.
65. OMG. Business Process Model and Notation, 2018. https://www.omg.org/spec/BPMN/2.0/.
66. D. A. Orchard and N. Yoshida. Effects as sessions, sessions as effects. In *POPL 2016*, pages 568–581, 2016.
67. L. Padovani. A simple library implementation of binary sessions. *J. Funct. Program.*, 27:e4, 2017.
68. D. A. Peled. Ten years of partial order reduction. In *CAV 1998*, pages 17–28, 1998.
69. W. Peng and S. Purushothaman. Analysis of a class of communicating finite state machines. *Acta Inf.*, 29(6/7):499–522, 1992.
70. R. Perera, J. Lange, and S. J. Gay. Multiparty compatibility for concurrent objects. In *PLACES 2016*, pages 73–82, 2016.
71. Introduction to protocol engineering. Available at http://cs.uccs.edu/~cs522/pe/pe.htm, 2006.
72. S. Qadeer and D. Wu. KISS: keep it simple and sequential. In *PLDI 2004*, pages 14–24, 2004.
73. G. Salaün, L. Bordeaux, and M. Schaerf. Describing and reasoning on web services using process algebra. *IJBPIM*, 1(2):116–128, 2006.
74. A. Scalas, O. Dardha, R. Hu, and N. Yoshida. A linear decomposition of multiparty sessions for safe distributed programming. In *ECOOP 2017*, pages 24:1–24:31, 2017.
75. A. Scalas and N. Yoshida. Lightweight session programming in scala. In *ECOOP 2016*, pages 21:1–21:28, 2016.
76. Scribble Project homepage, 2018. www.scribble.org.
77. K. C. Sivaramakrishnan, M. Qudeisat, L. Ziarek, K. Nagaraj, and P. Eugster. Efficient sessions. *Sci. Comput. Program.*, 78(2):147–167, 2013.
78. K. Takeuchi, K. Honda, and M. Kubo. An interaction-based language and its typing system. In *PARLE 1994*, pages 398–413, 1994.
79. R. Taylor, E. Tuosto, N. Walkinshaw, and J. Derrick. Choreography-based analysis of distributed message passing programs. In *PDP 2016*, pages 512–519, 2016.

80. D. M. Yellin and R. E. Strom. Protocol specifications and component adaptors. *ACM Trans. Program. Lang. Syst.*, 19(2):292–333, 1997.

## A Overview of the proofs of Lemma 1 and Theorem 5

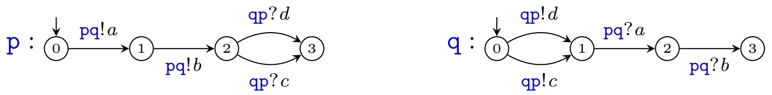
The properties $k$-OBI and IBI, and $k$-exhaustivity *together* guarantee that any choice made by an automaton is not constrained nor influenced by the channel bounds. The proof that $k$-MC guarantees safety for such systems crucially relies on this. The independence of choice wrt. the channel bounds for these CSA allows us to construct sets of executions that include all possible individual choices. We characterise this form of closure with the definition below, which is crucial for the further developments of this section.

**Definition 25 ($k$-Closed).** *Given a system $S$, $\Psi \subseteq \mathcal{A}^*$, and $s \in RS_k(S)$, we say that $\Psi$* is $k$-closed for $s$*, if the following two conditions hold:*

1. $\forall \phi \in \Psi : \exists s' \in RS_k(S) : s \xrightarrow{\phi}_k s'$
2. $\forall \phi_0 \cdot \mathtt{pq}!a \cdot \phi_1 \in \Psi$ *s.t.* $s \xrightarrow{\phi_0} (\boldsymbol{q}; \boldsymbol{w})$ *and* $\forall (q_{\mathtt{p}}, \ell, q'_{\mathtt{p}}) \in \delta_{\mathtt{p}}$ *there is* $\phi_0 \cdot \phi_2 \cdot \ell \cdot \phi_3 \in \Psi$ *with* $\phi_2 \cdot \phi_3 \in \mathcal{A}^*$ *and* $\mathtt{p} \notin \phi_2$.

In other words, $\Psi$ is $k$-closed for $s$ if (1) all executions in $\Psi$, starting from $s$, lead to a configuration in $RS_k(S)$ and (2) whenever an automaton $\mathtt{p}$ fires a send action in an execution in $\Psi$, then all possible choices that $\mathtt{p}$ can make are also represented in $\Psi$.

*Example 15.* Consider the 1-MC system $(M_{\mathtt{p}}, M_{\mathtt{q}})$ below.



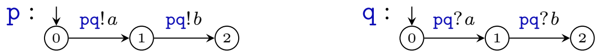
The sets $\{\epsilon\}$ and $\{\mathtt{qp}!c, \mathtt{qp}!d, \epsilon\}$ are both 1-closed for $s_0 = (0, 0; \epsilon, \epsilon)$. Instead, the set $\{\mathtt{qp}!c, \epsilon\}$ is not 1-closed for $s_0$ since there is a branching in participant $\mathtt{q}$ that is not represented.

Lemma 8 follows from the facts that $(i)$ $S$ is (reduced) $k$-OBI and $(ii)$ $S$ is $k$-exhaustive, i.e., all send actions are eventually enabled within the $k$-bounded executions.

**Lemma 8.** *Let $S$ be* reduced $k$-OBI *and $k$-exhaustive. For all $s \in RS_k(S)$, if $s \xrightarrow{\mathtt{pq}!a}$ and $\Psi = \{\phi \mid s \xrightarrow{\phi}_k \xrightarrow{\mathtt{pq}!a}_k \land \mathtt{p} \notin \phi\}$, then $\Psi \neq \varnothing$ is $k$-closed for $s$.*

Note that if $\mathtt{pq}!a$ is the only action enabled at $s$, then $\Psi = \{\epsilon\}$. In general, we do not have $\epsilon \in \Psi$, as shown in the example below.

*Example 16.* Consider the 1-MC system $(M_{\mathtt{p}}, M_{\mathtt{q}})$ below.



Pose $s = (1, 0; a, \epsilon)$, we have that the set $\{\phi \mid s \xrightarrow{\phi}_1 \xrightarrow{\mathtt{pq}!b}_1 \land \mathtt{p} \notin \phi\} = \{\mathtt{pq}?a\}$ is 1-closed for $s$. Indeed, for the action $\mathtt{pq}!b$ to be fired in a 1-bounded execution, message $a$ must be consumed first.

Lemma 9 below states that if there is a $k$-closed set of executions for a configuration $s$, we can construct another $k$-closed set for any successor of $s$.

**Lemma 9.** *Let $S$ be a $k$-IBI system, $s, s' \in RS_k(S)$ and $\Psi \subseteq \mathcal{A}^*$ such that $\Psi$ is $k$-closed for $s$, $s \xrightarrow{\ell}_k s'$, and $\hat{\Psi} = \hat{\Psi}_1 \cup \hat{\Psi}_2$, where*

$$\hat{\Psi}_1 = \{\phi \mid \phi \in \Psi \wedge \mathit{subj}(\ell) \notin \phi\} \text{ and } \hat{\Psi}_2 = \{\phi_1 \cdot \phi_2 \mid \phi_1 \cdot \ell \cdot \phi_2 \in \Psi \wedge \mathit{subj}(\ell) \notin \phi_1\}$$

*Then the following holds:*

1. *The set $\hat{\Psi}$ is $k$-closed for $s'$*
2. *For all $\psi \in \hat{\Psi}$, there is $\phi \in \Psi$ such that either:*
   - *$\psi \in \hat{\Psi}_1$, $\psi = \phi$, $\mathit{subj}(\ell) \notin \psi$, and there are $t, t' \in RS_k(S)$ such that $s \xrightarrow{\psi}_k t$, $s' \xrightarrow{\psi}_k t'$, and $t \xrightarrow{\ell}_k t'$, and $\phi \cdot \ell \asymp \ell \cdot \psi$; or*
   - *$\psi \in \hat{\Psi}_2$, there is $t \in RS_k(S)$ such that $s \xrightarrow{\phi}_k t$, $s' \xrightarrow{\psi}_k t$, and $\phi \asymp \ell \cdot \psi$.*
3. *$\Psi = \varnothing \iff \hat{\Psi} = \varnothing$.*

Figure 9 (left and middle) illustrates the construction of the executions in $\hat{\Psi}$. The crucial part of the proof is to show that $\hat{\Psi}$ is indeed $k$-closed, this is done by case analysis on the structure of an arbitrary execution in $\hat{\Psi}$. The assumption that $S$ is a $k$-IBI system is key here: we can rely on the fact that if $\ell$ is a receive action, then it is the unique receive action that $\mathit{subj}(\ell)$ can execute from $s$.

Next, Lemma 10 states that given the existence of a $k$-closed set of executions, one can find an alternative but equivalent path to a common configuration. We show the result below by induction on $n$, using Lemma 9.

**Lemma 10.** *Let $S$ be a reduced $k$-OBI and $k$-IBI system, then for all $s_1, \ldots, s_n \in RS_k(S)$, such that $s_1 \xrightarrow{\ell_1}_k s_2 \cdots s_{n-1} \xrightarrow{\ell_{n-1}}_k s_n$ (with $n > 1$). If there is $\varnothing \neq \Psi \subseteq \mathcal{A}^*$ such that $\Psi$ is $k$-closed for $s_1$, then there is $\phi_1 \in \Psi$ and $\psi, \phi_n \in \mathcal{A}^*$ such that $s_1 \xrightarrow{\phi_1}_k t_1 \xrightarrow{\psi}_k t_n$ and $s_n \xrightarrow{\phi_n}_k t_n$, for some $t_1, t_n \in RS_k(S)$ with $|\psi| < n$ and $\phi_1 \cdot \psi \asymp \ell_1 \cdots \ell_n \cdot \phi_n$.*

Figure 9 (right) illustrates Lemma 10. A key consequence of Lemma 8 and Lemma 10 is that if $s_1 \in RS_k(S)$, then we have $s_1 \xrightarrow{\phi_1}_k t_1 \xrightarrow{\ell_1}_k$, i.e., $t_1 \in RS_k(S)$; we use this result to show Lemma 11.

**Lemma 11.** *Let $S$ be reduced $k$-OBI, $k{+}1$-IBI, and $k$-exhaustive, then for all $s \in RS_k(S)$ and $s' \in RS_{k+1}(S)$ such that $s \xrightarrow{\phi}_{k+1} s'$, there is $t \in RS_k(S)$ and $\psi$, $\psi' \in \mathcal{A}^*$, such that $s \xrightarrow{\psi}_k t$, $s' \xrightarrow{\psi'}_{k+1} t$, and $\psi \asymp \phi \cdot \psi'$.*

Lemma 11 states that if $S$ is (reduced) $k$-OBI, $k{+}1$-IBI, and $k$-exhaustive then there is a path from any $k{+}1$-reachable configuration to a $k$-reachable configuration. The proof is by induction on the length of $\phi$ using Lemma 8 as a starting assumption, then applying Lemma 10 repeatedly.

*Remark 5.* The assumption that $S$ is $k{+}1$-IBI is required, see Figure 2 for an example that is 1-OBI, 1-IBI, and 1-exhaustive but for which the conclusions of Lemma 11 do not hold.

$$\begin{tikzcd}
s \arrow[r, "\ell"] \arrow[d, "\psi = \phi"'] & s' \arrow[d, "\psi = \phi"] \\
t \arrow[r, "\ell"] & t'
\end{tikzcd}$$

$$\begin{tikzcd}
s \arrow[r, "\ell"'] \arrow[d, "\phi_1"'] & s' \arrow[dd, bend left, "\phi_1"] \\
\cdot \arrow[d, "\ell"'] & \\
\cdot \arrow[d, "\phi_2"'] & \\
t & \begin{array}{l} \psi = \phi_1 \cdot \phi_2 \\ \phi = \phi_1 \cdot \ell \cdot \phi_2 \end{array}
\end{tikzcd}$$

$$\begin{tikzcd}
s_1 \arrow[r, "\ell_1"] \arrow[d, "\phi_1"'] & s_2 \cdots\cdots s_{n-1} \arrow[r, "\ell_{n-1}"] & s_n \arrow[d, "\phi_n"] \\
t_1 \arrow[rr, "\psi"] & & t_n
\end{tikzcd}$$

**Fig. 9.** Illustrations for Lemma 9 and Lemma 10.

Since the IBI property is undecidable in general, we have introduced the $k$-CIBI and $k$-SIBI properties as sound approximations of IBI, for $k$-OBI and $k$-exhaustive systems. We give a brief overview of the proof of Lemma 12 (part of which implies Lemma 2). The proof that $k$-CIBI implies IBI is similar, see Lemma 30 for the key result.

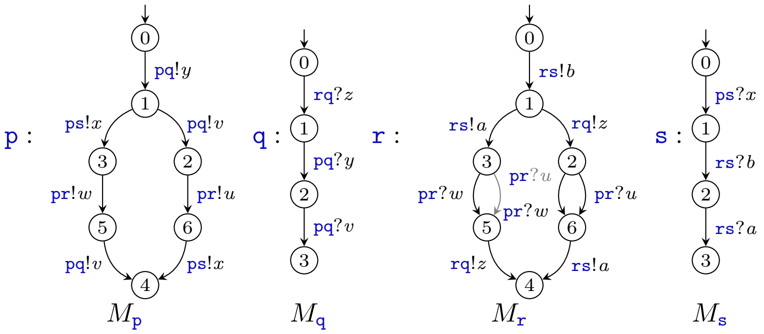
**Lemma 12.** *If $S$ is reduced $k$-*OBI*, $k$-*SIBI*, and $k$-exhaustive, then it is $k{+}1$-*SIBI*.*

To show Lemma 12, we show that for any system that is reduced $k$-OBI, $k$-SIBI, and $k$-exhaustive, the $k{+}1$-IBI property holds, i.e., Lemma 26. The proof of Lemma 26 is by induction on the length of an execution from $s_0$. Then we show the final result by contradiction, using Lemma 11 to find an execution that leads to a $k$-reachable configuration.

# B Additional examples

## B.1 Example for Section 3 — Behaviour of sending states

The system $S = (M_{\mathtt{p}}, M_{\mathtt{q}}, M_{\mathtt{r}}, M_{\mathtt{s}})$ (without the shaded part) is 1–MC but *not* $k$-safe (for any $k > 1$). Note that $M_{\mathtt{p}}$ and $M_{\mathtt{r}}$ are *not* send directed. The system with the shaded part is 1–MC and safe.

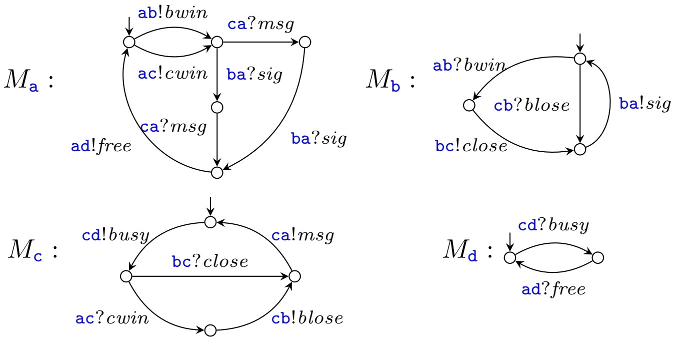


Interestingly, this example shows that Lemma 8 does not hold for non-$k$-OBI communicating automata. Take $s = (\boldsymbol{q}; \boldsymbol{w})$ such that $\boldsymbol{q} = (1, 0, 0, 0)$ and $w_{\mathtt{pq}} = y$ and the other channels are empty, then the set $\{\phi \mid s \xrightarrow{\phi}_1 \xrightarrow{\mathtt{pq}!v}_1 \wedge \mathtt{p} \notin \phi\}$ is *not*

1-closed for $s$. In particular, while participant $\mathtt{r}$ can execute $\mathtt{rs}!b \cdot \mathtt{rp}!z$ before $\mathtt{p}$ fires $\mathtt{pq}!v$, $\mathtt{r}$ cannot fire $\mathtt{rs}!b \cdot \mathtt{rs}!a$ (since the queue $\mathtt{rs}$ is full after firing $\mathtt{rs}!b$). This violates the definition of 1-closure since $\mathtt{r}$ can potentially send both $a$ and $z$ from state 1.

### B.2 Example for Section 3 — *k*-SIBI vs. *k*-CIBI

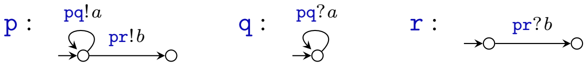
We illustrate the difference between the $k$-SIBI and $k$-CIBI properties with the system below. It is adapted from the running example of [51] where we have removed mixed states (choosing one interleaving for each outgoing transition). We refer to it as the 4 Player game in Table 2.



This system is $k$-IBI for all $k$ (and thus IBI): it is never the case that $M_{\mathtt{b}}$ (resp. $M_{\mathtt{c}}$) can choose between consuming *bwin* or *blose* (resp. *cwin* or *close*). It is not $k$-SIBI (for any $k$) because of the cyclic nature of the protocol (both choices are available at each iteration). However, this system is $k$-CIBI because, $M_{\mathtt{a}}$ need to receive acknowledgements from both $M_{\mathtt{b}}$ and $M_{\mathtt{c}}$ before starting a new iteration of the game; hence there is a dependency between, e.g., $\mathtt{ab}?\mathit{bwin}$ and $\mathtt{cb}!\mathit{blose}$.

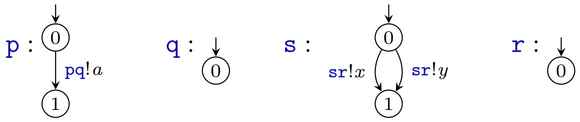
### B.3 Example for Section 4 — (reduced) *k*-OBI

The example below is *reduced* $k$-OBI for $k \geqslant 2$, but not $k$-OBI for any $k \geqslant 1$. $TS_1(S)$ includes a state where the queue $\mathtt{pq}$ contains one message $a$ and $M_{\mathtt{p}}$ is back and its initial state. At this point, $\mathtt{pr}!b$ is fireable, but $\mathtt{pq}!a$ is not. In $RTS_2(2)$, there is only one state from which $\mathtt{p}$ fires its send actions, both of which are enabled, hence the system is 2-OBI.



### B.4 Example for Section 4 — Ordered list

We illustrate the motivation to sort the list generated by *partition*(_), see Definition 15, with the system below.

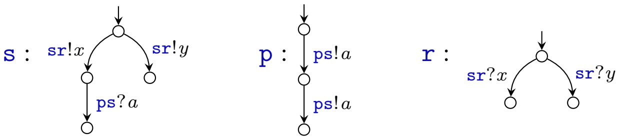


If we were to build the $RTS_k(S)$ of this system without sorting the list returned by $partition(s_0)$. We may obtain $partition(s_0) = \{\mathtt{sr}!x, \mathtt{sr}!y\} \cdot \{\mathtt{pq}!a\}$, which produces 4 transitions (and 5 states). Instead, if the list is sorted by ascending cardinality, we have $partition(s_0) = \{\mathtt{pq}!a\} \cdot \{\mathtt{sr}!x, \mathtt{sr}!y\}$, which gives us an $RTS_k(S)$ with 3 transitions (and 4 states).

*Remark 6.* Note that even though sorting sets of transitions by cardinality gives better performance in general, it does guarantee to find the smallest $RTS_k(S)$.

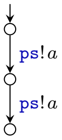
### B.5 Example for Section 3.2 — Local bound-agnosticity

We illustrate the reason for using projections which preserve $\epsilon$-transitions, i.e., $\pi^{\epsilon}_{\mathtt{p}}(TS_k(S))$, to characterise $k$-exhaustive systems, instead of projections which determinise the automata, cf. [51]. Consider the system $S$ below.

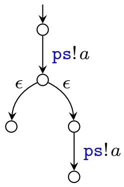


The traditional projections ($\pi_{\mathtt{p}}(TS_k(S))$) and projections ($\pi^{\epsilon}_{\mathtt{p}}(TS_k(S))$) for $k \in \{1, 2\}$ are given below (up to (weak) bisimulation).

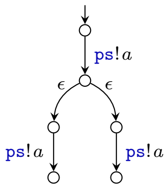
$\pi_{\mathtt{p}}(TS_1(S)) = \pi_{\mathtt{p}}(TS_2(S)) =$



$\pi^{\epsilon}_{\mathtt{p}}(TS_1(S)) =$



$\pi^{\epsilon}_{\mathtt{p}}(TS_2(S)) =$



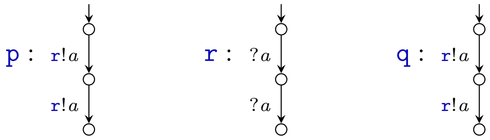
Observe that we have $\pi_{\mathtt{p}}(TS_1(S)) \sim \pi_{\mathtt{p}}(TS_2(S))$, but *not*

$$\pi^{\epsilon}_{\mathtt{p}}(TS_1(S)) \approx \pi^{\epsilon}_{\mathtt{p}}(TS_2(S))$$

Indeed, the system above is *not* 1-MC, but is 2-MC.

### B.6 Example for Sections 7 and 8 — Mailbox communicating automata

Consider the system $(M_{\mathtt{p}}, M_{\mathtt{r}}, M_{\mathtt{q}})$ below, with a mailbox semantics, i.e., participant $\mathtt{r}$ has one input queue to which both participants $\mathtt{p}$ and $\mathtt{q}$ can send messages.



If this system executes with bound $k \leqslant 3$, one participant (either $\mathtt{p}$ or $\mathtt{q}$) will be prevented to send at least one message. This namely implies that the send action of participant may become disabled after being enabled. This is problematic for the current partial order reduction algorithm and for the notion of $k$-closed sets used to prove our main results.

## C Proofs for Section 2

**Proposition 1.** *(1) If $S$ is send directed, then $S$ is $k$-OBI for all $k \in \mathbb{N}_{>0}$. (2) If $S$ is receive directed, then $S$ is IBI (and $k$-IBI for all $k \in \mathbb{N}_{>0}$).*

*Proof.* Immediate since each directed (CSA) automaton has access to at most one channel from each state. □

**Lemma 13.** *Let $S$ be a system and $\phi \in \mathcal{A}^*$. If $s_0 \xrightarrow{\phi}_k$, then $\phi$ is a* valid *execution.*

*Proof.* By induction on the length of $\phi$. The result follows trivially for $\phi = \epsilon$. Assume it holds for $\phi$ and let us show that is also holds for $\phi \cdot \ell$. Assume $\mathit{chan}(\ell) = \mathtt{pq}$. By induction hypothesis, for each prefix $\psi$ of $\phi$, we have that $\pi^{?}_{\mathtt{sr}}(\psi)$ is a prefix of $\pi^{!}_{\mathtt{sr}}(\psi)$ for any channel $\mathtt{sr} \in \mathcal{C}$. Hence, for each prefix $\psi$ of $\phi \cdot \ell$ we have that $\pi^{?}_{\mathtt{sr}}(\psi)$ is a prefix of $\pi^{!}_{\mathtt{sr}}(\psi)$ for any channel $\mathtt{sr} \neq \mathtt{pq} \in \mathcal{C}$. If $\ell = \mathtt{pq}!a$, the result still holds since $\pi^{!}_{\mathtt{sr}}(\psi)$ is longer or equal. The interesting case is when $\ell = \mathtt{pq}?a$. Pose $\pi^{!}_{\mathtt{pq}}(\phi) = \pi^{?}_{\mathtt{pq}}(\phi) \cdot w$ (there is such $w$ by induction hypothesis). Assume by contradiction that $\phi \cdot \mathtt{pq}?a$ is not a valid word. Then, there is no $w' \in \Sigma^*$ such that $\pi^{!}_{\mathtt{pq}}(\phi) = \pi^{!}_{\mathtt{pq}}(\phi \cdot \mathtt{pq}?a) = \pi^{?}_{\mathtt{pq}}(\phi \cdot \mathtt{pq}?a) \cdot w'$. which implies that either $w = b \cdot w''$ or $w = \epsilon$ $(b \neq a)$. This contradicts the fact that $s_0 \xrightarrow{\phi}_k s \xrightarrow{\mathtt{pq}?a}_k$ since the channel $\mathtt{pq}$ in $s$ is either empty or starts with $b$. □

**Lemma 14.** *Let $S$ be a system. If $s_0 \xrightarrow{\psi_0} s$, $s \xrightarrow{\phi} t$, and $s \xrightarrow{\phi'} t'$ such that $\phi \asymp \phi'$, then (1) $t = t'$ and (2) $\phi_0 \cdot \phi \Bumpeq \phi_0 \cdot \phi'$.*

*Proof.* Item (1) follows from the fact that the automata are deterministic hence, they all terminate in the same state, and the queues are consumed uniformly in both executions. Item (2) follows from the fact that both executions are valid, by Lemma 13. □

## D Proofs for Section 3

**Theorem 2.** *The problems of checking the $k$-*OBI*, $k$-*IBI*, $k$-*SIBI*, $k$-safety, and $k$-exhaustivity properties are all decidable and* PSPACE*-complete (with $k \in \mathbb{N}_{>0}$ given in unary). The problem of checking the $k$-*CIBI *property is decidable.*

*Proof.* We first observe that decidability follows straightforwardly since for any finite $k$, both $RS_k(S)$ and $\rightarrow_k$ are finite. We follow the proof of [11, Theorem 6.3]. Let $n$ be the maximum of $\{|Q_{\mathtt{p}}| \mid \mathtt{p} \in \mathcal{P}\}$, then there are at most $n|\mathcal{P}|$ local states in $S$.
**($k$-exhaustivity)** We check whether $S$ is *not* $k$-exhaustive, i.e., for each sending state $q_{\mathtt{p}}$ and send action from $q_{\mathtt{p}}$, we check whether there is a reachable configuration from which this send action cannot be fired. Hence, we need to search $RS_k(S)$, which has an exponential number of states (wrt. $k$). Following [11, Theorem 6.3], each configuration $s \in RS_k(S)$ may be encoded in space

$$|\mathcal{P}| \log n + |\mathcal{C}|k \log |\Sigma|$$

We also need one bit to remember whether we are looking for $q_{\mathtt{p}}$ or whether we are looking for the matching action. We need to store at most $|\mathcal{P}|n|\mathcal{C}||\Sigma|^k$ configurations, hence the problem can be decided in polynomial space when $k$ is given in unary.

Next, we show that the problem is PSPACE-hard. From [33, Proposition 5.5], we know that checking existentially stable $k$-boundedness for a system with the stable property is PSPACE-complete. By Theorem 8, this problem can be reduced to checking whether the system is $k$-exhaustive, which implies that checking $k$-exhaustivity must be PSPACE-hard.
**($k$-OBI)** For each sending state $q_{\mathtt{p}}$, we check whether there is a reachable configuration from which not all send actions can be fired, and thus we reason similarly to the $k$-exhaustivity case. Next, we show that checking $k$-OBI is PSPACE-hard. For this we adapt the construction from [15, Theorem 10] which reduces the problem of checking if the product of a set of finite state automata has an empty language to checking 1-synchronisability. We use the same construction as theirs (which is 1-OBI) but instead of adding states and transitions to ensure that the system breaks 1-synchronisability when each automata is in a final state, we add states and transitions that violate 1-OBI (using a construction like the one in Example 6).
**($k$-IBI)** For each non-directed receiving state $q_{\mathtt{p}}$, we check whether there is a reachable configuration from which more than one receive action can be fired, and thus we reason similarly as for $k$-exhaustivity. Showing that $k$-IBI is PSPACE-hard is similar to the $k$-OBI case.
**($k$-SIBI)** There are two components of this property, one is equivalent to $k$-IBI, the other requires to guarantee that no matching send action is fired from an already enabled receive state. Hence, for each non-directed receiving state $q_{\mathtt{p}}$, we check whether there is a reachable configuration from which one receive action of $\mathtt{p}$ is enabled, followed by a send action that matches another receive. We can proceed as in the case for $k$-exhaustivity with additional space to remember

whether we are looking for the receiving state or for a matching send action. Showing that $k$-SIBI PSPACE-hard is similar to the $k$-OBI case.
**($k$-safety)** For eventual reception, we proceed as in $k$-SIBI for each receiving state and element of the alphabet (check if such a configuration is reachable, then we search for a matching receive). For progress, we proceed as in $k$-SIBI for each receiving state $q_{\mathtt{p}}$ (check if such a configuration is reachable, then we search for a move by $\mathtt{p}$). Showing that checking $k$-safety PSPACE-hard is similar to the $k$-OBI case. □

**Lemma 15.** *Let $S$ s.t. $s \in RS_k(S)$ and $\Psi \subseteq \mathcal{A}^*$ such that $\Psi$ is $k$-closed for $s$, then $\Psi$ is $k{+}1$-closed for $s$.*

*Proof.* The result follows from Definition 25, since $\rightarrow_k \subseteq \rightarrow_{k+1}$. □

**Lemma 8.** *Let $S$ be* reduced *$k$-OBI and $k$-exhaustive. For all $s \in RS_k(S)$, if $s \xrightarrow{\mathtt{pq}!a}$ and $\Psi = \{\phi \mid s \xrightarrow{\phi}_k \xrightarrow{\mathtt{pq}!a}_k \wedge \mathtt{p} \notin \phi\}$, then $\Psi \neq \varnothing$ is $k$-closed for $s$.*

*Proof.* The non-emptiness of $\Psi$ follows easily from the assumption that $S$ is $k$-exhaustive (Definition 9). We have to show the following two conditions hold:
**(1)** $\forall \phi \in \Psi : \exists s' \in RS_k(S) : s \xrightarrow{\phi}_k s'$, which follows trivially from the definition of $\Psi$.
**(2)** For all $\phi_0 \cdot \mathtt{sr}!b \cdot \phi_1 \in \Psi$ such that $s \xrightarrow{\phi_0} (\boldsymbol{q}; \boldsymbol{w})$ and for all $(q_{\mathtt{s}}, \ell, q'_{\mathtt{s}}) \in \delta_{\mathtt{s}}$ there is $\phi_0 \cdot \ell \cdot \phi_2 \in \Psi$. For this part, take $\phi_0 \cdot \mathtt{sr}!b \cdot \phi_1 \in \Psi$ such that $s \xrightarrow{\phi_0} s' = (\boldsymbol{q}; \boldsymbol{w})$ (with $\mathtt{s} \neq \mathtt{p}$ by definition of $\Psi$). By definition of $\Psi$, we have $s' = (\boldsymbol{q}; \boldsymbol{w}) \in RS_k(S)$.

Since $S$ is $k$-exhaustive, for each $(q_{\mathtt{s}}, \mathtt{st}!c, q'_{\mathtt{s}}) \in \delta_{\mathtt{s}}$ there is $\psi$ s.t. we obtain the following situation (where each arrow indicates a $k$-bounded execution):

$$\begin{tikzcd}
s' \arrow[r, "\mathtt{sr}!b"] \arrow[d, "\psi"'] & {} \\
t \arrow[r, "\mathtt{st}!c"] & t'
\end{tikzcd}$$

with $\mathtt{s} \notin \psi$

There are two cases:

- If $\mathtt{p} \notin \psi$, we have that the local state of $\mathtt{p}$ in configurations $s$, $s'$ and $t$ is the same. Hence, by $k$-exhaustivity: $t' \xrightarrow{\psi'}_k \xrightarrow{\mathtt{pq}!a}_k$ with $\mathtt{p} \notin \psi$. Therefore, $\phi_0 \cdot \psi \cdot \mathtt{st}!c \cdot \psi' \in \Psi$ as required.
- If there is no $\phi$ such that $\mathtt{p} \notin \psi$, then there must be a dependency chain in $\psi$ that prevents $\mathtt{st}!c$ to be fired without $\mathtt{p}$ making a move. Since $s \notin \psi$, we must have some $\mathtt{st}?d$ in $\psi$ such that $\mathtt{st}?d$ depends on an action by $\mathtt{p}$. The smallest such chain is of the form: $\mathtt{pt}!x \cdot \mathtt{pt}?x \cdot \mathtt{st}?y$. Without loss of generality, pose $\psi = \mathtt{pt}!x \cdot \mathtt{pt}?x \cdot \mathtt{st}?y$ (we reason similarly with a longer chain).

  Take $\phi_3$ s.t. $s_0 \xrightarrow{\phi_3}_k s$, since $S$ is reduced $k$-OBI and $k$-exhaustive, there are $t''$ and $\psi_0$ such that $s_0 \xrightarrow{\psi_0}_k t''$, and $\phi_4$ s.t. $t' \xrightarrow{\phi_4}_k t''$, with

$$\psi_0 \asymp \phi_3 \cdot \phi_0 \cdot \mathtt{pt}!x \cdot \mathtt{pt}?x \cdot \mathtt{st}?y \cdot \mathtt{st}!c \cdot \phi_4$$

by Lemma 39 (2). Hence, due to the dependency chain within $\psi$, we must have:

$$\psi_0 = \psi_1 \cdot \mathtt{pt}!x \cdot \psi_2 \cdot \mathtt{pt}?x \cdot \psi_3 \cdot \mathtt{st}?y \cdot \psi_4 \cdot \mathtt{st}!c \cdot \psi_5$$

with $\mathtt{s} \notin \psi_2 \cdot \psi_3 \cdot \psi_4$. There are three cases:

- Either $\mathtt{sr}!b$ is $k$-enabled immediately after $\psi_1$, in which case we have a contradiction with the fact that $S$ is reduced $k$-OBI,
- $\mathtt{sr}!b$ is $k$-enabled strictly after $\psi_1$ and strictly before $\mathtt{st}!c$, then we have a contradiction with the fact that $S$ is reduced $k$-OBI, or
- $\mathtt{sr}!b$ is not $k$-enabled along $\psi_0$, which is also a contradiction with the fact that $S$ is reduced $k$-OBI. □

Given $\phi = \ell_1 \cdots \ell_n \in \mathcal{A}^*$, we write $subj(\phi)$ for the set $\bigcup_{1 \leqslant i \leqslant n}\{subj(\ell_i)\}$.

**Lemma 16.** *If $s \xrightarrow{\phi}_k t$ and $s \xrightarrow{\psi}_k t'$ and $subj(\phi) \cap subj(\psi) = \varnothing$, then there is $s'$ such that $t \xrightarrow{\psi}_k s'$ and $t' \xrightarrow{\phi}_k s'$.*

*Proof.* Straightforward: the executions are independent from one another. □

**Lemma 9.** *Let $S$ be a $k$-IBI system, $s, s' \in RS_k(S)$ and $\Psi \subseteq \mathcal{A}^*$ such that $\Psi$ is $k$-closed for $s$, $s \xrightarrow{\ell}_k s'$, and $\hat{\Psi} = \hat{\Psi}_1 \cup \hat{\Psi}_2$, where*

$$\hat{\Psi}_1 = \{\phi \mid \phi \in \Psi \wedge subj(\ell) \notin \phi\} \text{ and } \hat{\Psi}_2 = \{\phi_1 \cdot \phi_2 \mid \phi_1 \cdot \ell \cdot \phi_2 \in \Psi \wedge subj(\ell) \notin \phi_1\}$$

*Then the following holds:*

1. *The set $\hat{\Psi}$ is $k$-closed for $s'$*
2. *For all $\psi \in \hat{\Psi}$, there is $\phi \in \Psi$ such that either:*
   - *$\psi \in \hat{\Psi}_1$, $\psi = \phi$, $subj(\ell) \notin \psi$, and there are $t, t' \in RS_k(S)$ such that $s \xrightarrow{\psi}_k t$, $s' \xrightarrow{\psi}_k t'$, and $t \xrightarrow{\ell}_k t'$, and $\phi \cdot \ell \asymp \ell \cdot \psi$; or*
   - *$\psi \in \hat{\Psi}_2$, there is $t \in RS_k(S)$ such that $s \xrightarrow{\phi}_k t$, $s' \xrightarrow{\psi}_k t$, and $\phi \asymp \ell \cdot \psi$.*
3. *$\Psi = \varnothing \iff \hat{\Psi} = \varnothing$.*

*Proof.* Let us pose $subj(\ell) = \mathtt{p}$.
**(1)** We first observe that $\hat{\Psi}$ validates condition (1) of Definition 25, i.e., $\forall \phi \in \hat{\Psi} : \exists s'' \in RS_k(S) : s' \xrightarrow{\phi}_k s''$, by definition of $\hat{\Psi}$. We then show that $\hat{\Psi}$ validates the second condition of $k$-closure. There are two cases depending on whether the execution is in $\hat{\Psi}_1$ or $\hat{\Psi}_2$.

1. Take $\phi = \phi_0 \cdot \mathtt{sr}!a \cdot \phi_1 \in \hat{\Psi}_1$, then by definition of $\hat{\Psi}_1$, we have $\mathtt{p} \neq \mathtt{s}$ and $\phi \in \Psi$. Hence, posing $s \xrightarrow{\phi_0} (\boldsymbol{q}; \boldsymbol{w})$, we have that for all $(q_{\mathtt{s}}, \ell', q'_{\mathtt{s}}) \in \delta_{\mathtt{s}}$ there is $\phi_0 \cdot \phi_1 \cdot \ell' \cdot \phi_2 \in \Psi$, with $subj(\ell') \notin \phi_1$, since $\Psi$ is $k$-closed by assumption.
   (a) If $\mathtt{p} \notin \phi_2$, then $\phi_0 \cdot \phi_1 \cdot \ell' \cdot \phi_2 \in \hat{\Psi}_1$, as required.
   (b) If $\mathtt{p} \in \phi_2$, then there are two cases depending on whether $\ell$ is a send or a receive action.
      - If $\ell = \mathtt{qp}?a$, then we must have $\phi_2 = \phi_3 \cdot \mathtt{qp}?a \cdot \phi_4$ with $\mathtt{p} \notin \phi_3$, since $S$ is $k$-IBI (only one receive action can be enabled at $\mathtt{p}$). Thus $\phi_0 \cdot \phi_1 \cdot \ell' \cdot \phi_3 \cdot \phi_4 \in \hat{\Psi}_2$, as required.

- If $\ell = \mathtt{pq}!a$, then we must have $\phi_2 = \phi_3 \cdot \mathtt{pt}!b \cdot \phi_4$ with $\mathtt{p} \notin \phi_3$. Since $\varPsi$ is $k$-closed, we also have $\phi_0 \cdot \phi_1 \cdot \ell' \cdot \phi_3 \cdot \phi_4 \cdot \mathtt{pq}!a \cdot \phi_5 \in \varPsi$,
  for some $\phi_4, \phi_5$ s.t. $\mathtt{p} \notin \phi_4$. Thus, $\phi_0 \cdot \phi_1 \cdot \ell' \cdot \phi_3 \cdot \phi_4 \cdot \phi_5 \in \hat{\varPsi}_2$, as required.

2. Take $\phi = \phi_0 \cdot \mathtt{sr}!a \cdot \phi_1 \in \hat{\varPsi}_2$. There are two cases:
   (a) If $\phi_0 = \phi_2 \cdot \phi_3$ and $\phi_2 \cdot \ell \cdot \phi_3 \cdot \mathtt{sr}!a \cdot \phi_1 \in \varPsi$, then posing $s \xrightarrow{\phi_2 \cdot \ell \cdot \phi_3} (\boldsymbol{q}; \boldsymbol{w})$, we have that for all $(q_{\mathtt{s}}, \ell', q'_{\mathtt{s}}) \in \delta_{\mathtt{s}}$ there is $\phi_2 \cdot \ell \cdot \phi_3 \cdot \phi_5 \cdot \ell' \cdot \phi_4 \in \varPsi$ (for some $\phi_4$ and $\phi_5$ s.t. $\mathtt{s} \notin \phi_5$) since $\varPsi$ is $k$-closed by assumption. Thus, $\phi_2 \cdot \phi_3 \cdot \phi_5 \cdot \ell' \cdot \phi_4 \in \hat{\varPsi}_2$, as required.
   (b) If $\phi_1 = \phi_2 \cdot \phi_3$ and $\phi_0 \cdot \mathtt{sr}!a \cdot \phi_2 \cdot \ell \cdot \phi_3 \in \varPsi$, then $\mathtt{p} \notin \phi_0 \cdot \mathtt{sr}!a \cdot \phi_2$ (hence $\mathtt{p} \neq \mathtt{s}$) and, posing $s \xrightarrow{\phi_0} (\boldsymbol{q}; \boldsymbol{w})$, we have that for all $(q_{\mathtt{s}}, \ell', q'_{\mathtt{s}}) \in \delta_{\mathtt{s}}$ there is $\phi_0 \cdot \phi_8 \cdot \ell' \cdot \phi_4 \in \varPsi$ (for some $\phi_4$ and $\phi_8$ s.t. $\mathtt{s} \notin \phi_8$) since $\varPsi$ is $k$-closed by assumption.
   - if $\mathtt{p} \notin \phi_4$, then $\phi_0 \cdot \phi_8 \cdot \ell' \cdot \phi_4 \in \hat{\varPsi}_1$, as required.
   - if $\mathtt{p} \in \phi_4$, there are two cases depending on whether $\ell$ is a receive or send action.
     - if $\ell$ is a receive action, then we must have $\phi_4 = \phi_5 \cdot \ell \cdot \phi_6$ with $\mathtt{p} \notin \phi_5$, thus $\phi_0 \cdot \phi_8 \cdot \ell' \cdot \phi_5 \cdot \phi_6 \in \hat{\varPsi}_2$, as required, since $S$ is $k$-IBI (only one receive action can be enabled at $\mathtt{p}$)
     - if $\ell$ is a send action, pose $\ell = \mathtt{pq}!c$, then we must have $\phi_4 = \phi_5 \cdot \mathtt{pt}!b \cdot \phi_6$ with $\mathtt{p} \notin \phi_5$. Since $\varPsi$ is $k$-closed, we must also have $\phi_0 \cdot \ell' \cdot \phi_5 \cdot \phi_9 \cdot \mathtt{pq}!c \cdot \phi_7 \in \varPsi$ (for some $\phi_7$ and $\phi_9$ s.t. $\mathtt{p} \notin \phi_9$). Thus, $\phi_0 \cdot \ell' \cdot \phi_5 \cdot \phi_9 \cdot \phi_7 \in \hat{\varPsi}_2$, as required.

**(2)** Take $\psi \in \hat{\varPsi}$, by definition of $\hat{\varPsi}$, there are two cases:

1. If $\psi \in \hat{\varPsi}_1$, then $\psi = \phi \in \varPsi$ and since $subj(\ell) \notin \psi$, $s \xrightarrow{\psi}_k t \xrightarrow{\ell}_k t'$ and $s' \xrightarrow{\psi}_k t'$ by Lemma 16. In picture, we have

$$\begin{tikzcd}
s \arrow[r, "\ell"] \arrow[d, "\psi = \phi"'] & s' \arrow[d, "\psi = \phi"] \\
t \arrow[r, "\ell"] & t'
\end{tikzcd}$$

   Finally, we have $\phi \cdot \ell \asymp \ell \cdot \psi$ since $subj(\ell) \notin \psi$.
2. If $\psi \in \hat{\varPsi}_2$, then there is $\phi = \phi_0 \cdot \ell \cdot \phi_1 \in \varPsi$ s.t. $\psi = \phi_0 \cdot \phi_1$ and $subj(\ell) \notin \phi_0$. Thus, by Lemma 16 we have $s \xrightarrow{\phi_0 \cdot \ell \cdot \phi_1}_k t$ and $s \xrightarrow{\ell}_k s' \xrightarrow{\phi_0 \cdot \phi_1}_k t$, i.e.,

$$\begin{tikzcd}
s \arrow[r, "\ell"] \arrow[d, "\phi_0"'] & s' \arrow[ddl, bend left, "\phi_0"] \\
\cdot \arrow[d, "\ell"'] & \\
\cdot \arrow[d, "\phi_1"'] & \\
t &
\end{tikzcd}$$

Finally, we have $\phi_0 \cdot \ell \cdot \phi_1 \asymp \ell \cdot \phi_0 \cdot \phi_1$ since $subj(\ell) \notin \phi_0$.

**(3)** The ($\Rightarrow$) direction is trivial from the definition of $\hat{\Psi}$. Let us show that $\hat{\Psi} = \varnothing \implies \Psi = \varnothing$ by contradiction. Assume $\hat{\Psi} = \varnothing$ and $\Psi \neq \varnothing$. This implies that for all $\phi \in \Psi : \mathtt{p} \in \phi$. Pose $\phi = \phi_0 \cdot \hat{\ell} \cdot \phi_1$, with $\mathtt{p} \notin \phi_0$, $\ell \neq \hat{\ell}$.

- If $\ell$ is a receive action, then $\hat{\ell}$ is also a receive action ($\mathtt{p} \notin \phi_0$), thus $\ell \neq \hat{\ell}$ contradicts the assumptions that $s \xrightarrow{\ell}_k$ and $\mathtt{p} \notin \phi_0$.
- If $\ell$ is a send action, then $\hat{\ell}$ is also a send action ($\mathtt{p} \notin \phi_0$), thus it is a contradiction with the fact that $\Psi$ is $k$-closed for $s$. □

**Lemma 10.** *Let $S$ be a reduced $k$-*OBI *and $k$-*IBI *system, then for all $s_1, \dots, s_n \in RS_k(S)$, such that $s_1 \xrightarrow{\ell_1}_k s_2 \cdots s_{n-1} \xrightarrow{\ell_{n-1}}_k s_n$ (with $n > 1$). If there is $\varnothing \neq \Psi \subseteq \mathcal{A}^*$ such that $\Psi$ is $k$-closed for $s_1$, then there is $\phi_1 \in \Psi$ and $\psi, \phi_n \in \mathcal{A}^*$ such that $s_1 \xrightarrow{\phi_1}_k t_1 \xrightarrow{\psi}_k t_n$ and $s_n \xrightarrow{\phi_n}_k t_n$, for some $t_1, t_n \in RS_k(S)$ with $|\psi| < n$ and $\phi_1 \cdot \psi \asymp \ell_1 \cdots \ell_n \cdot \phi_n$.*

*Proof.* By replicated application of Lemma 9 (parts 1 and 3), for all $1 \leqslant i \leqslant n$, there is $\varnothing \neq \Psi_i \subseteq \mathcal{A}^*$ such that $\Psi_i$ is $k$-closed for $s_i$. In addition, by Lemma 9 (part 2), for all $1 \leqslant i < n$, and for all $\phi_{i+1} \in \Psi_{i+1}$, there is $\phi_i \in \Psi_i$ such that either

- $s_{i+1} \xrightarrow{\phi_{i+1}}_k t_{i+1}$, and $s_i \xrightarrow{\phi_i}_k t_i$, with $t_i = t_{i+1}$, or
- $s_{i+1} \xrightarrow{\phi_{i+1}}_k t_{i+1}$, $s_i \xrightarrow{\phi_i}_k t_i$, and $t_i \xrightarrow{\ell_i}_k t_{i+1}$.

The rest of the proof is by induction on $n$.

**(Base case)** If $n = 2$, then the result follows directly by instantiating Lemma 9 with $s_1 = s$, $s_n = s'$, and $\ell_1 = \ell$, in particular, we have $\psi = \ell_1$ or $\psi = \epsilon$ (hence $|\psi| < n$).

**(Inductive case)** Assume the result holds for $n = i$ (i.e., $\phi_1 \cdot \psi \asymp \ell_1 \cdots \ell_{i-1} \cdot \phi_i$) and let us show that it holds for $n = i+1$. We have the following situation:

$$\begin{tikzcd}
s_1 \arrow[r, dashed, "\ell_1 \cdots \ell_{i-1}"] \arrow[d, "\phi_1"] & s_i \arrow[r, "\ell_i"] \arrow[d, "\phi_i"] & s_{i+1} \arrow[d, "\phi_{i+1}"] \\
t_1 \arrow[r, dashed, "\psi"] & t_i \arrow[r, "\psi'"] & t_{i+1}
\end{tikzcd}$$

By Lemma 9, we have either

1. $t_i = t_{i+1}$, $\psi' = \epsilon$, and $\phi_i \cdot \epsilon \asymp \ell_i \cdot \phi_{i+1}$.
2. $\psi' = \ell_i$, $\phi_i = \phi_{i+1}$ and $\phi_i \cdot \ell_i \asymp \ell_i \cdot \phi_i$.

We have to show that

$$\phi_1 \cdot \psi \cdot \psi' \asymp \ell_1 \cdots \ell_{i-1} \cdot \ell_i \cdot \phi_{i+1}$$

- Assume case (1) holds.

$$\begin{aligned}
\phi_1 \cdot \psi &\asymp \ell_1 \cdots \ell_{i-1} \cdot \phi_i && \text{by induction hypothesis}\\
&\asymp \ell_1 \cdots \ell_{i-1} \cdot \phi' \cdot \ell_i \cdot \phi'' && \text{posing } \phi_i = \phi' \cdot \ell_i \cdot \phi'' \text{ with } \mathit{subj}(\ell_i) \notin \phi'\\
&\asymp \ell_1 \cdots \ell_{i-1} \cdot \ell_i \cdot \phi' \cdot \phi'' && \text{since } \mathit{subj}(\ell_i) \notin \phi'\\
&\asymp \ell_1 \cdots \ell_{i-1} \cdot \ell_i \cdot \phi_{i+1} && \text{by Lemma 9}
\end{aligned}$$

  Finally, since $\psi' = \epsilon$ in this case, we have $\phi_1 \cdot \psi \cdot \psi' = \phi_1 \cdot \psi$, hence

$$\phi_1 \cdot \psi \asymp \ell_1 \cdots \ell_{i-1} \cdot \ell_i \cdot \phi_{i+1}$$

  as required.
- Assume case (2) holds.

$$\begin{aligned}
\phi_1 \cdot \psi &\asymp \ell_1 \cdots \ell_{i-1} \cdot \phi_i && \text{by induction hypothesis}\\
\phi_1 \cdot \psi \cdot \ell_i &\asymp \ell_1 \cdots \ell_{i-1} \cdot \phi_i \cdot \ell_i && \text{by Lemma 14}\\
&\asymp \ell_1 \cdots \ell_{i-1} \cdot \ell_i \cdot \phi_i && \text{by case (2)}\\
&\asymp \ell_1 \cdots \ell_{i-1} \cdot \ell_i \cdot \phi_{i+1} && \text{by case (2)}\\
\phi_1 \cdot \psi \cdot \psi' &\asymp \ell_1 \cdots \ell_{i-1} \cdot \ell_i \cdot \phi_{i+1} && \psi' = \ell_i
\end{aligned}$$

In both cases, we have $|\psi \cdot \psi'| \leqslant i$ since $|\psi| < i$ by induction hypothesis and $\phi = \epsilon$ (resp. $\psi' = \ell_i$) by case (1) (resp. case (2)). □

**Lemma 11.** *Let $S$ be reduced $k$-OBI, $k$+1-IBI, and $k$-exhaustive, then for all $s \in RS_k(S)$ and $s' \in RS_{k+1}(S)$ such that $s \xrightarrow{\phi}_{k+1} s'$, there is $t \in RS_k(S)$ and $\psi$, $\psi' \in \mathcal{A}^*$, such that $s \xrightarrow{\psi}_k t$, $s' \xrightarrow{\psi'}_{k+1} t$, and $\psi \asymp \phi \cdot \psi'$.*

*Proof.* We show the result by induction on the length of $\phi$.
**(Base case)** If $\phi = \epsilon$, then the result holds trivially with $s = s' = t = t' \in RS_k(S)$.
**(Inductive case)** Assume that for all $s \in RS_k(S)$ and $s' \in RS_{k+1}(S)$ such that $s \xrightarrow{\phi}_{k+1} s'$, with $|\phi| < n$, there is $t \in RS_k(S)$ and $\psi$, $\psi' \in \mathcal{A}^*$, such that $s \xrightarrow{\psi}_k t$, $s' \xrightarrow{\psi'}_{k+1} t$, and $\psi \asymp \phi \cdot \psi'$.

Take $s \in RS_k(S)$ and $s' \in RS_{k+1}(S)$ such that $s \xrightarrow{\phi}_{k+1} s'$, with $\phi = \ell_1 \cdots \ell_n$ (i.e., $|\phi| = n$), assuming that

$$s = s_1 \xrightarrow{\ell_1}_{k+1} s_2 \xrightarrow{\ell_2}_{k+1} \cdots \xrightarrow{\ell_n}_{k+1} s_{n+1} = s'$$

There are two cases depending on the direction of $\ell_1$.

1. If $\ell_1 = \mathtt{pq}?a$, then $s_2 \in RS_k(S)$ since $s_1 \in RS_k(S)$. Thus, by induction hypothesis, there is $t \in RS_k(S)$ and $\psi$, $\psi' \in \mathcal{A}^*$, such that $s_2 \xrightarrow{\psi}_k t$ and $s' \xrightarrow{\psi'}_{k+1} t$ and $\psi \asymp \ell_2 \cdots \ell_n \cdot \psi'$. Hence, $\ell_1 \cdot \psi \asymp \ell_1 \cdot \ell_2 \cdots \ell_n \cdot \psi'$, as required since $s_1 \xrightarrow{\ell_1}_k s_2$.

2. If $\ell_1 = \mathtt{pq}!a$, then by Lemma 8, the set $\Psi_1 = \{\psi \mid s \xrightarrow{\psi}_k \xrightarrow{\mathtt{pq}!a}_k \wedge \mathtt{p} \notin \phi\}$ is non-empty and $\Psi_1$ is $k$-closed for $s$.
   Therefore, by Lemma 15, $\Psi_1$ is $k{+}1$-closed for $s$ and by Lemma 9, the set

   $$\Psi_2 = \{\phi \mid \phi \in \Psi_1 \wedge \mathit{subj}(\ell_1) \notin \phi\} \cup \{\phi_1 \cdot \phi_2 \mid \phi_1 \cdot \ell_1 \cdot \phi_2 \in \Psi_1 \wedge \mathit{subj}(\ell_1) \notin \phi_1\}$$

   is $k{+}1$-closed for $s_2$ and $\Psi_1 = \Psi_2 = \{\phi \mid \phi \in \Psi_1 \wedge \mathit{subj}(\ell_1) \notin \phi\}$ by definition of $\Psi_1$.
   Hence, since $S$ is $k{+}1$-BI by assumption, we can apply Lemma 10 and obtain that there is $\psi_2 \in \Psi_2$ and $\hat{\phi}'$, $\psi_{n+1} \in \mathcal{A}^*$ such that

   $$s_2 \xrightarrow{\psi_2}_{k+1} t_2 \xrightarrow{\hat{\phi}'}_{k+1} t_{n+1} \quad \text{and} \quad s' = s_{n+1} \xrightarrow{\psi_{n+1}}_k t_{n+1}$$

   for some $t_2, t_{n+1} \in RS_{k+1}(S)$ with $|\hat{\phi}| < n$ and

   $$\psi_2 \cdot \hat{\phi}' \asymp \ell_2 \cdots \ell_n \cdot \psi_{n+1}$$

   We have $t_2 \xrightarrow{\hat{\phi}'}_{k+1} t_{n+1}$, with $|\hat{\phi}'| < n$, with $t_2 \in RS_k(S)$, thus by induction hypothesis, there is $t \in RS_k(S)$ such that $t_{n+1} \xrightarrow{\hat{\psi}'}_{k+1} t$, $s_2 \xrightarrow{\hat{\psi}}_k t$ and $\hat{\psi} \asymp \hat{\phi}' \cdot \hat{\psi}'$, as pictured below (where red parts are in $\rightarrow_k$ and the rest in $\rightarrow_{k+1}$).

   $$\begin{tikzcd}
   s = s_1 \arrow[r, "\ell_1"] \arrow[d, "\psi_1 = \psi_2"'] & s_2 \arrow[r, "\ell_2 \cdots \ell_n"] \arrow[d, "\psi_2"] & s_{n+1} = s' \arrow[d, "\psi_{n+1}"] \\
   t_1 \arrow[r, "\ell_1"] & t_2 \arrow[r, "\hat{\phi}'"] \arrow[dr, dashed, "\hat{\psi}"'] & t_{n+1} \arrow[d, dashed, "\hat{\psi}'"] \\
   & & t
   \end{tikzcd}$$

   We have to show that

   $$\psi_1 \cdot \ell_1 \cdot \hat{\psi} \asymp \ell_1 \cdots \ell_n \cdot \psi_{n+1} \cdot \hat{\psi}'$$

   By Lemma 10,

   $$\psi_1 \cdot \hat{\phi}' \asymp \ell_2 \cdots \ell_n \cdot \psi_{n+1}$$

   Prefixing each execution with $\ell_1$, we have:

   $$\ell_1 \cdot \psi_1 \cdot \hat{\phi}' \asymp \ell_1 \cdot \ell_2 \cdots \ell_n \cdot \psi_{n+1}$$

   and since $\mathit{subj}(\ell_1) \notin \psi_1$, we have:

   $$\psi_1 \cdot \ell_1 \cdot \hat{\phi}' \asymp \ell_1 \cdot \ell_2 \cdots \ell_n \cdot \psi_{n+1}$$

   Adding $\hat{\psi}'$ on each side of the equation, we obtain:

   $$\psi_1 \cdot \ell_1 \cdot \hat{\phi}' \cdot \hat{\psi}' \asymp \ell_1 \cdot \ell_2 \cdots \ell_n \cdot \psi_{n+1} \cdot \hat{\psi}'$$

By induction hypothesis, we have $\hat{\psi} \asymp \hat{\phi}' \cdot \hat{\psi}'$. Hence, we obtain

$$\psi_1 \cdot \ell_1 \cdot \hat{\psi} \asymp \ell_1 \cdots \ell_n \cdot \psi_{n+1} \cdot \hat{\psi}'$$

as required. □

**Lemma 17.** *If $S$ is reduced $k$-*OBI*, $k{+}1$-*IBI*, and $k$-exhaustive, then for all $s \in RS_{k+1}(S)$, there is $t \in RS_k(S)$ such that $s \rightarrow^*_{k+1} t$.*

*Proof.* Direct consequence of Lemma 11. □

**Lemma 18.** *If $S$ is reduced $k$-*OBI*, $(k{+}1)$-*IBI*, and $k$-exhaustive, then it is reduced $(k{+}1)$-*OBI* and $(k{+}1)$-exhaustive.*

*Proof.* **($(k{+}1)$-OBI)** By contradiction, assume $S$ is reduced $k$-OBI but not reduced $(k{+}1)$-OBI. Then, there must be $s = (\boldsymbol{q}; \boldsymbol{w}) \in RTS_{k+1}(S) \backslash RTS_k(S)$ such that there is $\mathtt{p} \in \mathcal{P}$, $s \xrightarrow{\mathtt{pr}!b}_{k+1}$, $(q_{\mathtt{p}}, \mathtt{pr}!b, q'_{\mathtt{p}}) \in \delta_{\mathtt{p}}$, and $\neg(s \xrightarrow{\mathtt{pt}!c}_{k})$. By Lemma 17 and Lemma 39 (2), there is $t' \in RTS_k(S)$ such that $s \xrightarrow{\phi}_{k+1} t'$. There are two cases:

1. If $\mathtt{pt}?x \notin \psi_1$, then we have $t' \xrightarrow{\mathtt{pr}!b}_{k+1}$, and $\neg(t' \xrightarrow{\mathtt{pt}!c}_{k+1})$, hence $\neg(t' \xrightarrow{\mathtt{pt}!c}_{k})$.
   - If $t' \xrightarrow{\mathtt{pr}!b}_{k}$ we have a contradiction with the fact that $S$ is reduced $k$-OBI.
   - If $\neg(t' \xrightarrow{\mathtt{pr}!b}_{k})$ then both queues are full at $t'$. Since $S$ is $k$-exhaustive, both actions are enabled along a $k$-bounded execution from $t'$. However, one action must be enabled before the other, in any execution, contradicting the fact that $S$ is reduced $k$-OBI.
2. If $\mathtt{pt}?x \in \psi_1$, $t' \xrightarrow{\mathtt{pr}!b}_{k+1}$, and $t' \xrightarrow{\mathtt{pt}!c}_{k+1}$. Then the queue $\mathtt{pt}$ must still be holding $k$ messages at $t'$. Hence, $\neg(t' \xrightarrow{\mathtt{pt}!c}_{k})$ and we reason as above to reach a contradiction with the fact that $S$ is reduced $k$-OBI.

**($(k{+}1)$-exhaustive)** By contradiction, assume $S$ is $k$-exhaustive, but not $(k{+}1)$-exhaustive. Then, there must be $s = (\boldsymbol{q}; \boldsymbol{w}) \in RS_{k+1} \backslash RS_k(S)$ such that there is $\mathtt{p} \in \mathcal{P}$, with $q_{\mathtt{p}}$ a sending state and the following does *not* hold:

$$\forall (q_{\mathtt{p}}, \mathtt{pq}!a, q'_{\mathtt{p}}) \in \delta_{\mathtt{p}} : \exists \phi \in \mathcal{A}^* : s \xrightarrow{\phi}_{k+1} \xrightarrow{\mathtt{pq}!a}_{k+1} \text{ and } \mathtt{p} \notin \phi \tag{1}$$

By Lemma 17, there is $s' \in RS_k(S)$ such that $s \xrightarrow{\phi}_{k+1} s'$.

1. If $\mathtt{p} \notin \phi$, then $s' \xrightarrow{\phi'}_{k} \xrightarrow{\mathtt{pq}!a}_{k}$ (by $k$-MC and $s' \in RS_k(S)$), i.e., a contradiction.
2. If $\mathtt{p} \in \phi$. There are two cases:
   (a) $\phi = \phi_1 \cdot \mathtt{pq}!a \cdot \phi_2$ with $\mathtt{p} \notin \phi_1$, hence $\mathtt{pq}!a$ can be fired from $s$, a contradiction with the assumption that (D) above does not hold.
   (b) $\phi = \phi_1 \cdot \mathtt{pt}!b \cdot \phi_2$ with $\mathtt{p} \notin \phi_1$ and $a \neq b$. This implies that

$$s \xrightarrow{\phi_1}_{k+1} \xrightarrow{\mathtt{pq}!a}_{k+1} \text{ since } S \text{ is BI}$$

which contradicts the assumption that (D) does not hold. □

**Lemma 19.** *If $S$ is (reduced) $k$-OBI, IBI, and $k$-exhaustive, then for all $s \in RS(S)$ such that $s_0 \xrightarrow{\phi} s$, there is $t \in RS_k(S)$ and $\psi$, $\phi' \in \mathcal{A}^*$, such that $s_0 \xrightarrow{\psi}_k t$, $s \xrightarrow{\phi'} t$, and $\psi \asymp \phi \cdot \phi'$.*

*Proof.* We first note that in this case $\Bumpeq$ and $\asymp$ coincide since we only consider executions starting from $s_0$, see Lemma 13; thus we show that $\psi \Bumpeq \phi \cdot \phi'$.

From Lemma 18, we know that $S$ is $n$-exhaustive (for any $n \geqslant k$). Hence, we obtain the result by repeated applications of Lemma 11 (with $s = s_0$) using the fact that $\Bumpeq$ is a congruence. □

**Lemma 1.** *If $S$ is $k$-OBI, IBI, and $k$-MC, then it is $k$+1-OBI and $(k$+1$)$-MC.*

*Proof.* By Lemma 20 and Lemma 3. □

**Lemma 20.** *If $S$ is reduced $k$-OBI, IBI, and $k$-MC then it is $k$+1-OBI and $(k$+1$)$-MC.*

*Proof.* Assume by contradiction, that $S$ is $k$-MC, but not $(k$+1$)$-safe. Then, there must be $s = (\boldsymbol{q}; \boldsymbol{w}) \in RS_{k+1} \backslash RS_k(S)$ such that at least one of the following conditions does *not* hold.

1. For all $\mathtt{pq} \in \mathcal{C}$, if $w_{\mathtt{pq}} = a \cdot w'$, then $s \rightarrow_{k+1}^{*} \xrightarrow{\mathtt{pq}?a}_{k+1}$.
2. For all $\mathtt{p} \in \mathcal{P}$, if $q_{\mathtt{p}}$ is a *receiving* state, then $s \rightarrow_{k+1}^{*} \xrightarrow{\mathtt{qp}?a}_{k+1}$ for some $\mathtt{q} \in \mathcal{P}$ and $a \in \Sigma$.

Note that $S$ is $(k$+1$)$-OBI and $k$+1-exhaustive by Lemma 18.

By Lemma 17, there is $s' \in RS_k(S)$ such that $s \xrightarrow{\phi}_{k+1} s'$.

**(1)** Assume that Item 1 above does not hold, i.e., we have $w_{\mathtt{pq}} = a \cdot w'$ for some $\mathtt{pq} \in \mathcal{C}$, but each path $\phi$ from $s$ does not contain $\mathtt{pq}?a$. Observe that for the first occurrence of $\mathtt{pq}?b$ in $\phi$, we must have $a = b$ (since $w_{\mathtt{pq}} = a \cdot w'$), but we cannot have $\mathtt{pq}?a \in \phi$ by contradiction hypothesis. This implies that we have $w'_{\mathtt{pq}} = a \cdot w' \cdot w''$ in $s'$, and since $S$ is $k$-MC and $s' \in RS_k(S)$, we must have $s' \rightarrow_k{}^* \xrightarrow{\mathtt{pq}?a}_k$. Thus, we have $s \xrightarrow{\phi}_{k+1} s' \rightarrow_k{}^* \xrightarrow{\mathtt{pq}?a}_k$, a contradiction.

**(2)** Assume that Item 2 above does not hold, i.e., there is $\mathtt{p} \in \mathcal{P}$ such that $q_{\mathtt{p}}$ is a receiving state but for each path $\phi$ from $s$, $\phi$ does not allow $\mathtt{p}$ to fire a (receive) action. Hence, by contradiction hypothesis we have $\mathtt{qp}?a \notin \phi$ for any $a$ and $\mathtt{q}$. Hence $\mathtt{p}$ is still in state $q_{\mathtt{q}}$ in configuration $s'$. Since $S$ is $k$-MC and $s' \in RS_k(S)$, we must have $s' \rightarrow_k{}^* \xrightarrow{\mathtt{qp}?a}_k$. Thus, we have $s \xrightarrow{\phi}_{k+1} s' \rightarrow_k{}^* \xrightarrow{\mathtt{qp}?a}_k$, a contradiction. □

**Theorem 1.** *If $S$ is $k$-OBI, IBI, and $k$-MC, then it is safe.*

*Proof.* By Theorem 5 and Lemma 3. □

**Theorem 5.** *If $S$ is reduced $k$-OBI, IBI, and $k$-MC, then it is safe.*

*Proof.* Direct consequence of Lemma 20. □

**Lemma 21.** *Let $S$ be (reduced) $k$-OBI and IBI. If $S$ is safe and $k$-exhaustive, then it is $k$-MC.*

*Proof.* We show that $S$ is $k$-safe. By contradiction, assume there is $S$ safe, $k$-exhaustive, and *not* $k$-safe. Since $S$ is not $k$-safe, then there is $s = (\boldsymbol{q}; \boldsymbol{w}) \in RS_k(S)$ such that at least one of the two cases below hold.

1. $w_{\texttt{pq}} = a \cdot w$ and there is no execution $\phi$ such that $s \xrightarrow{\phi}_k \xrightarrow{\texttt{pq}?a}_k$. By safety, there is $\psi$ and $n > k$ such that $s \xrightarrow{\psi}_n s' \xrightarrow{\texttt{pq}?a}_n s''$. By Lemma 11, we can extend $\psi \cdot \texttt{pq}?a$ such that there is an equivalent $k$-bounded execution, which contradicts this case.
2. $q_{\texttt{p}}$ is a receiving state and there is no execution $\phi$ such that $s \xrightarrow{\phi}_k \xrightarrow{\texttt{qp}?a}_k$; then we reason similarly as above using Lemma 11. □

**Lemma 22.** *If $S$ is $k$-SIBI, then it is $k$-IBI.*

*Proof.* Straightforward. □

**Lemma 23.** *If $S$ is $k$-CIBI, then it is $k$-IBI.*

*Proof.* Straightforward. □

**Lemma 24.** *If $s \vdash \ell \prec_{\phi} \ell'$, then there is a subsequence $\psi$ of $\phi$ such that*

- *$s \vdash \ell \prec \ell'$ and $\psi = \epsilon$, or*
- *$\psi = \ell_1 \cdots \ell_n$ ($n \geqslant 1$), $s \vdash \ell \prec \ell_1$, $\forall 1 \leqslant i < n : s \vdash \ell_i \prec \ell_{i+1}$, $s \vdash \ell_n \prec \ell'$.*

*Proof.* By induction on the length of $\phi$.
**(Base case)** If $s \vdash \ell \prec_{\epsilon} \ell'$, then we must have $s \vdash \ell \prec \ell'$ by definition.
**(Inductive case)** Assume the result holds for $\phi$ and let us show it holds for $\ell'' \cdot \phi$. There are two cases:

- If $s \vdash \ell \prec_{\phi} \ell'$ and we have the result by induction hypothesis, since any subsequence of $\phi$ is a subsequence of $\ell'' \cdot \phi$.
- If $s \vdash \ell \prec \ell''$ and $s \vdash \ell'' \prec_{\phi} \ell'$. Then by induction hypothesis there is a subsequence $\ell_1 \cdots \ell_n$ of $\phi$ such that $\ell'' \prec \ell_1 \prec \cdots \ell_n \prec \ell'$ hence we have the result with the subsequence $\ell'' \cdot \ell_1 \cdots \ell_n$. □

**Lemma 25.** *Let $S$ be a system, $s \in RS(S)$, and $\phi = \phi_1 \cdot \ell \cdot \phi_2 \cdot \ell' \cdot \phi_3$ such that $s_0 \xrightarrow{\phi}$ and $s \vdash \ell \prec_{\phi_2} \ell'$, with $s_0 \xrightarrow{\phi_1} s$. Then for all valid $\psi$ such that $\psi \Bumpeq \phi$, there are $\psi_1$, $\psi_2$, $\psi_3$, and $t \in RS(S)$ such that*

1. *$\psi = \psi_1 \cdot \ell \cdot \psi_2 \cdot \ell' \cdot \psi_3$,*
2. *$\pi_{subj(\ell)}(\psi_1) = \pi_{subj(\ell)}(\phi_1)$*
3. *$\pi_{subj(\ell')}(\psi_1 \cdot \ell \cdot \psi_2) = \pi_{subj(\ell')}(\phi_1 \cdot \ell \cdot \phi_2)$, and*
4. *$t \vdash \ell \prec_{\psi_2} \ell'$, with $s_0 \xrightarrow{\psi_1} t$.*

*Proof.* By Lemma 24, there is a subsequence $\ell_1 \cdots \ell_n$ of $\phi_2$ such that

$$s \vdash \ell = \ell_0 \prec \ell_1 \text{ and } \forall 1 \leqslant i < n : s \vdash \ell_i \prec \ell_{i+1} \text{ and } s \vdash \ell_n \prec \ell' = \ell_{n+1}$$

Take the shortest such subsequence (smallest $n$), we show that the relative order between each pair of actions must be preserved. By definition, for each $s \vdash \ell_j \prec \ell_{j+1}$ $(0 \leqslant j \leqslant n+1)$ to hold there are two cases:

- If $\mathit{subj}(\ell_j) = \mathit{subj}(\ell_{j+1})$, then it is not possible to swap $\ell_j$ and $\ell_{j+1}$ while preserving $\Bumpeq$-equivalence.
- If $\mathit{subj}(\ell_j) \neq \mathit{subj}(\ell_{j+1})$, then $\mathit{chan}(\ell_j) = \mathit{chan}(\ell_{j+1})$, and there are two cases depending on whether the queue $\mathit{chan}(\ell_j)$ is empty when $\ell_j$ is fired.
  - If the queue is empty, then we cannot swap $\ell_j$ and $\ell_{j+1}$ without invalidating the execution since they are matching send and receive actions.
  - If the queue is not empty, since $w_{\mathit{chan}(\ell_j)} = \epsilon$ (at $s$) there must be another *send* action $\ell_l$ with $l < j$ such that $\mathit{chan}(\ell_l) = \mathit{chan}(\ell_{j+1})$. Therefore, we have $s \vdash \ell_l \prec \ell_{j+1}$, and thus $\ell_1 \cdots \ell_l \cdots \ell_{j+1} \cdots \ell_n$ is a (striclty) shorter subsequence of $\phi_2$ which is dependency chain, a contradiction.

Since each pair of actions cannot be swapped without invalidating the sequence or break $\Bumpeq$-equivalence, we must conclude that any $\psi$ has the required form and that the $t \vdash \ell \prec_{\psi_2} \ell'$ property holds since $\psi_2$ must contain the subsequence $\ell_1 \cdots \ell_n$. □

**Lemma 26.** *If $S$ is reduced $k$-*OBI*, $k$-*SIBI *and $k$-exhaustive, then it is $k$+1-*IBI.

*Proof.* From Lemma 27 and 28. □

**Lemma 27.** *If $S$ is $k$-*SIBI*, then it is $k$-*CIBI.

*Proof.* By contradiction, take $s = (\boldsymbol{q}; \boldsymbol{w}) \in RS_k(S)$ such that the condition for $k$-CIBI do not hold while the condition for $k$-SIBI does. Then, we must have $s \xrightarrow{\mathtt{qp}?a}_k s'$ and $s' \xrightarrow{\phi}_k \xrightarrow{\mathtt{sp}!b}_k$ such that $\neg(s \vdash \mathtt{qp}?a \prec_\phi \mathtt{sp}!b)$. However, the existence of an execution $s' \xrightarrow{\phi}_k \xrightarrow{\mathtt{sp}!b}_k$ contradicts Definition 12. □

**Lemma 28.** *If $S$ is reduced $k$-*OBI*, $k$-*CIBI *and $k$-exhaustive, then it is $k$+1-*IBI.

*Proof.* Take $s \in RS_k(S)$ and $s' \in RS_{k+1}(S)$ such that $s \xrightarrow{\phi}_{k+1} s'$. We show by induction on the length of $\phi$ that $s' \xrightarrow{\phi'}_{k+1} t'$ for some $t' \in RS_k(S)$, and there is $\psi$ such that $s \xrightarrow{\psi}_k t'$ with $\psi \asymp \phi \cdot \phi'$, and for all prefix $\phi'_0$ of $\phi'$, if $s' \xrightarrow{\phi'_0}_{k+1} s'' = (\boldsymbol{q}; \boldsymbol{w})$, $s''$ validates the following condition, for all $\mathtt{p} \in \mathcal{P}$:

$$s'' \xrightarrow{\mathtt{qp}?a}_{k+1} t \implies \forall \ell \in \mathcal{A} : s \xrightarrow{\ell}_{k+1} \wedge\, \mathit{subj}(\ell) = \mathtt{p} \implies \ell = \mathtt{qp}?a$$

**(Base case)** Assume $\phi = \ell$. If $\ell = \mathtt{pq}?a$, then $s' \in RS_k(S)$, and we have result since $S$ is $k$-CIBI (via Lemma 23), with $s' = t'$. If $\ell = \mathtt{pq}!a$, then since $S$ is $k$-exhaustive, we have $s \xrightarrow{\psi}_k t \xrightarrow{\mathtt{pq}!a}_k t'$, with $\mathtt{p} \notin \psi$. Hence, we have $s' \xrightarrow{\psi}_{k+1} t'$.

We show that for all prefix $\psi_0$ of $\psi$, if $s' \xrightarrow{\psi_0}_{k+1} t''$, then $t''$ validates the $k{+}1$-IBI condition. We have the following situation:

$$\begin{tikzcd}
s \arrow[r, "\ell = \texttt{pq}!a"] \arrow[d, "\psi_0"'] & s' \arrow[d, "\psi_0"] \\
s'' \arrow[r, "\ell"] \arrow[d, "\psi_1"'] & t'' \arrow[d, "\psi_1"] \\
t \arrow[r, "\ell"] & t'
\end{tikzcd}$$

Assume by contradiction that $t'' \xrightarrow{\texttt{sr}?b}_{k+1}$ and $t'' \xrightarrow{\texttt{tr}?c}_{k+1}$. If these two transitions are also enabled at $s''$, we have a contradiction with the fact that $S$ is $k$-CIBI. Hence, we have that either participant $\texttt{r}$ has made a move through $\ell$, hence $\texttt{p} = \texttt{r}$, an additional receive action in $\texttt{r}$ becomes enabled because $\texttt{sr} = \texttt{pq}$, or $\texttt{tr} = \texttt{pq}$ (i.e., the queue $\texttt{sr}$ (resp. $\texttt{tr}$) is empty in $s$ and $s''$).

- If $\texttt{p} = \texttt{r}$, then if we pose $\psi_0 = \psi$, we have $t' \xrightarrow{\texttt{sr}?b}_{k+1}$ and $t' \xrightarrow{\texttt{tr}?c}_{k+1}$, which contradicts the fact that $S$ is $k$-CIBI.
- If $\texttt{sr} = \texttt{pq}$ (i.e., $\texttt{sr}?b = \texttt{pq}?a$), then we have $s'' \xrightarrow{\texttt{tq}?c}_{k} v$ for some $v$. Since $S$ is $k$-exhaustive, we also have $v \xrightarrow{\psi_2}_{k} \xrightarrow{\texttt{pq}!a}_{k}$ with $\texttt{p} \notin \psi_2$. By $k$-CIBI, we have that for all such $\psi_2$, we have $s'' \vdash \texttt{tq}?c \prec_{\psi_2} \texttt{pq}!a$, which is a contradiction with Lemma 25 since the two actions are swapped in $k{+}1$.
- The case $\texttt{tr} = \texttt{pq}$ is symmetric to the one above.

**(Inductive case)** Assume the result holds for $\phi$ and let us show it holds for $\phi \cdot \ell$. Assume that we have the following situation, where the dashed edges need to be shown to exist.

$$\begin{tikzcd}
s \arrow[r, "\phi"] \arrow[dr, bend right, "\psi"'] & s' \arrow[r, "\ell"] \arrow[d, "\phi'"'] & s'' \arrow[d, dashed] \\
 & t' \arrow[r, dashed] & t''
\end{tikzcd}$$

with $s, t' \in RS_k(S)$ and $s', s'' \in RS_{k+1}(S)$.

By induction hypothesis, all configurations between $s'$ and $t'$ and between $s'$ and $s''$ are $k{+}1$-IBI and $k{+}1$-OBI (by Lemma 20), hence, we can use a similar reasoning to that of Lemma 9 to show that either $s'' \xrightarrow{\phi'}_{k+1} t''$ (with $t' \xrightarrow{\ell}_{k+1} t''$) or $s'' \xrightarrow{\phi'}_{k+1} t'$ (with $t' = t''$).

- If $s'' \xrightarrow{\phi'}_{k+1} t''$ (with $t' \xrightarrow{\ell}_{k+1} t''$), then we proceed as in the base case with $s := t'$ and $s' := t''$.

- If $s'' \xrightarrow{\phi'}_{k+1} t'$ (with $t' = t''$), then we only have to show that all configurations on $\phi'$ validate the condition. Since there is an equivalent $k$-bounded execution, any violation would contradict the hypothesis that $S$ is $k$-CIBI. □

**Lemma 12.** *If $S$ is reduced $k$-*OBI*, $k$-*SIBI*, and $k$-exhaustive, then it is $k{+}1$-*SIBI*.*

*Proof.* We note that since $S$ is reduced $k$-OBI, $k$-SIBI and $k$-exhaustive, we have that $S$ is $k{+}1$-IBI by Lemma 26. We show this result by contradiction, using Lemma 26 and Lemma 11. Assume, by contradiction, that there is $s \in RS_k(S)$ and $s' = (\boldsymbol{q}; \boldsymbol{w}) \in RS_{k+1}(S)$ such that $s \xrightarrow{\phi}_{k+1} s'$ with $\mathtt{p} \in \mathcal{P}$ s.t.

1. $s' \xrightarrow{\mathtt{qp}?a}_{k+1}$, and $s' \xrightarrow{\mathtt{sp}?b}_{k+1}$, or
2. $s' \xrightarrow{\mathtt{qp}?a}_{k+1}$, and $\exists (q_{\mathtt{p}}, \mathtt{sp}?b, q'_{\mathtt{p}}) \in \delta_{\mathtt{p}} : \mathtt{s} \neq \mathtt{q} \wedge s \rightarrow^{*}_{k+1} \xrightarrow{\mathtt{sp}!b}_{k+1}$

(1) follows from Lemma 26.

(2) Assume there is $s'$ such that $s' \xrightarrow{\mathtt{qp}?a}_{k+1}$, and $\exists (q_{\mathtt{p}}, \mathtt{sp}?b, q'_{\mathtt{p}}) \in \delta_{\mathtt{p}} : \mathtt{s} \neq \mathtt{q} \wedge s \xrightarrow{\phi'}_{k+1} \xrightarrow{\mathtt{sp}!b}_{k+1} s''$. By Lemma 11, there is $t \in RS_k(S)$ such that $s \xrightarrow{\psi}_{k} t$ and $s'' \xrightarrow{\phi'}_{k+1} t$ with $\psi \asymp \phi \cdot \phi \cdot \mathtt{sp}!b \cdot \phi''$. Hence both $\mathtt{qp}?a$ and $\mathtt{sp}!b$ appear in $\psi$ which contradicts the fact that $S$ is $k$-SIBI. □

**Lemma 29.** *If $S$ is $k$-*OBI*, $k$-*SIBI* and $k$-exhaustive, then it is* IBI.

*Proof.* Direct consequence of Lemma 12, Lemma 18, and Lemma 3. □

**Lemma 30.** *If $S$ is reduced $k$-*OBI*, $k$-*CIBI*, and $k$-exhaustive, then it is $k{+}1$-*CIBI*.*

*Proof.* We first note that since $S$ is reduced $k$-OBI, $k$-CIBI and $k$-exhaustive, we have that $S$ is $k{+}1$-IBI by Lemma 28.

We show this result by contradiction, using Lemma 28 and Lemma 11. Assume, by contradiction, that there is $s \in RS_k(S)$ and $s' = (\boldsymbol{q}; \boldsymbol{w}) \in RS_{k+1}(S)$ such that $s \xrightarrow{\phi}_{k+1} s'$ with $\mathtt{p} \in \mathcal{P}$, $(q_{\mathtt{p}}, \mathtt{sp}?b, q'_{\mathtt{p}}) \in \delta_{\mathtt{p}}$ and $\mathtt{s} \neq \mathtt{q}$ s.t.

1. $s' \xrightarrow{\mathtt{qp}?a}_{k+1}$, and $s' \xrightarrow{\mathtt{sp}?b}_{k+1}$, or
2. $s' \xrightarrow{\mathtt{qp}?a}_{k+1} s''$, $s'' \xrightarrow{\phi'}_{k+1} \xrightarrow{\mathtt{sp}!b}_{k+1} t$, and $\neg(s' \vdash \mathtt{qp}?a \prec_{\phi'} \mathtt{sp}!b)$

(1) is a contradiction with Lemma 28.

(2) By Lemma 11, there is $t' \in RS_k(S)$ such that $s \xrightarrow{\psi}_{k} t'$ and $t \xrightarrow{\phi''}_{k+1} t'$ with $\psi \asymp \phi \cdot \mathtt{qp}?a \cdot \phi' \cdot \mathtt{sp}!b \cdot \phi''$. There are two cases:

1. If $\psi = \psi_1 \cdot \mathtt{sp}!b \cdot \psi_2 \cdot \mathtt{qp}?a \cdot \psi_3$, with $\pi_{\mathtt{p}}(\psi_1 \cdot \mathtt{sp}!b \cdot \psi_2) = \pi_{\mathtt{p}}(\phi)$ and $\pi_{\mathtt{s}}(\psi_1) = \pi_{\mathtt{s}}(\phi \cdot \mathtt{qp}?a \cdot \phi')$, then we have a contradiction with the assumption that $S$ is $k$-CIBI since $\mathtt{p}$ can receive $b$ and $a$ after having executed $\pi_{\mathtt{p}}(\psi_1 \cdot \mathtt{sp}!b \cdot \psi_2)$, i.e., both messages are in the queue.

2. If $\psi = \psi_1 \cdot \mathtt{qp}?a \cdot \psi_2 \cdot \mathtt{sp}!b \cdot \psi_3$, with $\pi_{\mathtt{p}}(\psi_1) = \pi_{\mathtt{p}}(\phi)$ and $\pi_{\mathtt{s}}(\psi_1 \cdot \mathtt{qp}?a \cdot \psi_2) = \pi_{\mathtt{s}}(\phi \cdot \mathtt{qp}?a \cdot \phi')$, then we must have $\hat{s} \vdash \mathtt{qp}?a \prec_{\psi_2} \mathtt{sp}!b$ (assuming $s_0 \xrightarrow{\psi_1} \hat{s}$) since $S$ is $k$-CIBI. By Lemma 25, we must also have $s' \vdash \mathtt{qp}?a \prec_{\phi'} \mathtt{sp}!b$, a contradiction. □

**Lemma 31.** *If $S$ is $k$-OBI, $k$-CIBI and $k$-exhaustive, then it is IBI.*

*Proof.* By Lemma 18, Lemma 30, and Lemma 3. □

**Lemma 2.** *If $S$ is $k$-OBI, $k$-CIBI (resp. $k$-SIBI) and $k$-exhaustive, then it is IBI.*

*Proof.* By Lemma 31 and Lemma 29. □

### D.1 Completeness characterisation of $k$-exhaustive systems

**Lemma 32.** *If $S$ is (reduced) $k$-OBI, IBI, and $k$-exhaustive, then $\forall \mathtt{p} \in \mathcal{P} : \pi^{\epsilon}_{\mathtt{p}}(TS_k(S)) \approx \pi^{\epsilon}_{\mathtt{p}}(TS_{k+1}(S))$.*

*Proof.* Pose $TS_k(S) = (N, s_0, \varDelta)$ and $TS_{k+1}(S) = (N', s_0, \varDelta')$. Recall that we have $\varDelta \subseteq \varDelta'$ and $N \subseteq N'$.

Assume by contradiction that $\neg(\pi^{\epsilon}_{\mathtt{p}}(TS_k(S)) \approx \pi^{\epsilon}_{\mathtt{p}}(TS_{k+1}(S)))$ for some $\mathtt{p} \in \mathcal{P}$. Then, there are $s \in N \cap N'$ and $\ell$ (with $subj(\ell) = \mathtt{p}$) such that $s \xrightarrow{\phi}_{k+1} s' \xrightarrow{\ell}_{k+1} s''$ with $\pi_{\mathtt{p}}(\phi) = \epsilon$ and

$$\forall \phi' \in \mathcal{A} : \forall s'' \in RS_k(S) : s \xrightarrow{\phi'}_k s'' \land \pi_{\mathtt{p}}(\phi') = \epsilon \implies \neg(s'' \xrightarrow{\ell}_k) \tag{2}$$

By Lemma 11, there is $t \in RS_k(S)$ and $\psi$, $\psi' \in \mathcal{A}^*$, such that $s \xrightarrow{\psi}_k t$, $s'' \xrightarrow{\psi'}_{k+1} t$, $\psi \asymp \phi \cdot \ell \cdot \psi'$. Hence, we have $s \xrightarrow{\psi}_k$ with $\pi_{\mathtt{p}}(\psi) = \ell \cdot \psi''$ for some $\psi''$ with contradicts (D.1). □

**Lemma 33.** *If $S$ is such that $\exists k \in \mathbb{N}_{>0} : \forall \mathtt{p} \in \mathcal{P} : \pi^{\epsilon}_{\mathtt{p}}(TS_k(S)) \approx \pi^{\epsilon}_{\mathtt{p}}(TS_{k+1}(S))$, then $S$ is $k$-exhaustive.*

*Proof.* Assume by contradiction that there is some $k \in \mathbb{N}_{>0}$ such that

$$\forall \mathtt{p} \in \mathcal{P} : \pi^{\epsilon}_{\mathtt{p}}(TS_k(S)) \approx \pi^{\epsilon}_{\mathtt{p}}(TS_{k+1}(S)) \tag{3}$$

and $S$ is *not* $k$-exhaustive.

Pose $TS_k(S) = (N, s_0, \varDelta)$ and $TS_{k+1}(S) = (N', s_0, \varDelta')$. Recall that we have $\varDelta \subseteq \varDelta'$ and $N \subseteq N'$.

Since $S$ is *not* $k$-exhaustive, there are $s = (\boldsymbol{q}; \boldsymbol{w}) \in RS_k(S)$ and $\mathtt{pq} \in \mathcal{C}$ such that $s \xrightarrow{\mathtt{pq}!a}$ and

$$\forall \phi \in \mathcal{A}^* : \forall s' \in RS_k(S) : s \xrightarrow{\phi}_k s' \land \mathtt{p} \notin \phi \implies \neg(s' \xrightarrow{\mathtt{pq}!a}_k) \tag{4}$$

Since $s \in RS_k(S)$ and $\neg(s \xrightarrow{\mathtt{pq}!a}_k)$, we must have $|w_{\mathtt{pq}}| = k$. Hence, $s \xrightarrow{\mathtt{pq}!a}_{k+1}$ and therefore

$$(s, \mathtt{pq}!a, s'') \in \Delta' \qquad \text{for some } s'' \in N' \tag{5}$$

By (D.1) and the fact that $\Delta \subseteq \Delta'$ and $N \subseteq N'$, we must have

$$\pi^{\epsilon}_{\mathtt{p}}((N, s, \Delta)) \approx \pi^{\epsilon}_{\mathtt{p}}((N', s, \Delta'))$$

which is clearly a contradiction with (D.1) and (D.1). □

**Corollary 1.** *Let $S$ be $k$-*OBI *and* IBI *such that:*
$\exists k \in \mathbb{N}_{>0} : \forall \mathtt{p} \in \mathcal{P} : \pi^{\epsilon}_{\mathtt{p}}(TS_k(S)) \approx \pi^{\epsilon}_{\mathtt{p}}(TS_{k+1}(S))$.
*Then,* $\forall n \geqslant k : \forall \mathtt{p} \in \mathcal{P} : \pi^{\epsilon}_{\mathtt{p}}(TS_k(S)) \approx \pi^{\epsilon}_{\mathtt{p}}(TS_n(S))$.

*Proof.* Take $S$ such that $\exists k : \forall \mathtt{p} \in \mathcal{P} : \pi^{\epsilon}_{\mathtt{p}}(TS_k(S)) \approx \pi^{\epsilon}_{\mathtt{p}}(TS_{k+1}(S))$. Then, by Lemma 33, $S$ is $k$-exhaustive. Since $S$ is $k$-OBI and $(k+1)$-IBI by assumption, $S$ is $n$-exhaustive for any $n \geqslant k$, by Lemma 18. Hence, by Lemma 32, we have $\forall \mathtt{p} \in \mathcal{P} : \pi^{\epsilon}_{\mathtt{p}}(TS_n(S)) \approx \pi^{\epsilon}_{\mathtt{p}}(TS_{n+1}(S))$ (for any $n \geqslant k$). □

**Theorem 3.** *Let $S$ be a system.*

*(1) If* $\exists k \in \mathbb{N}_{>0} : \forall \mathtt{p} \in \mathcal{P} : \pi^{\epsilon}_{\mathtt{p}}(TS_k(S)) \approx \pi^{\epsilon}_{\mathtt{p}}(TS_{k+1}(S))$*, then $S$ is $k$-exhaustive.*
*(2) If $S$ is $k$-*OBI, IBI*, and $k$-exhaustive, then* $\forall \mathtt{p} \in \mathcal{P} : \pi^{\epsilon}_{\mathtt{p}}(TS_k(S)) \approx \pi^{\epsilon}_{\mathtt{p}}(TS_{k+1}(S))$.

*Proof.* Part (1) follows from Lemma 33 and Part (2) follows from Lemma 32. □

## E Proofs for Section 4

Below, we say that a configuration $s \in RS_k(S)$ is $k$-OBI (resp. $k$-IBI) if it validates the corresponding condition, e.g., if $\mathtt{p}$ can fire one send action from $s$, then all its send actions are enabled. We say that $S$ (resp. $s$) is $k$-BI when it is $k$-OBI and $k$-IBI.

**Definition 26.** *We say that $S$ is* reduced $k$-chained input bound independent *(reduced $k$-*CIBI*) if for all $s = (\boldsymbol{q}; \boldsymbol{w}) \in RS_k(S)$ and for all $\mathtt{p} \in \mathcal{P}$, if $s \xrightarrow{\mathtt{qp}?a}_k s'$, then* $\forall (q_{\mathtt{p}}, \mathtt{sp}?b, q'_{\mathtt{p}}) \in \delta_{\mathtt{p}} : \mathtt{s} \neq \mathtt{q} \implies \neg(s \xrightarrow{\mathtt{sp}?b}_k) \land (\forall \phi \in \mathcal{A}^* : s' \xrightarrow{\phi}_k \xrightarrow{\mathtt{sp}!b}_k \implies s \vdash \mathtt{qp}?a \prec_{\phi} \mathtt{sp}!b)$.

**Lemma 3.** *Let $S$ be a system, if $S$ is $k$-*OBI*, then $S$ is also reduced $k$-*OBI.

*Proof.* By contradiction. Notice that Definition 17 requires the same property than Definition 6 at the configuration level. Take $s \in \hat{N}$ s.t. $s$ violates the (reduced) $k$-OBI condition, then $s \in RS_k(S)$, and $s$ also violates $k$-OBI. □

**Lemma 34.** *Let $S$ be a system, if $S$ is $k$-*SIBI*, then $S$ is also* reduced $k$-SIBI.

*Proof.* By contradiction. Take $s \in \hat{N}$ s.t. it violates the (reduced) $k$-SIBI condition. Note that we $s \in RS_k(S)$. There are two cases:

- If there is $\mathtt{p}$ such that two receive actions are enabled for $\mathtt{p}$, then they are also enabled at $s$, a contradiction.
- If there is $\mathtt{p}$ such that one receive action is enabled for $\mathtt{p}$, and there is $\rightarrow_k$-path s.t. a conflicting send action is fired, then we have the situation in $TS_k(S)$, hence we have a contradiction. □

**Lemma 35.** *Let $S$ be a system, if $S$ is $k$-*CIBI*, then $S$ is also* reduced *$k$-*CIBI*.*

*Proof.* By contradiction. Take $s \in \hat{N}$ s.t. it violates the (reduced) $k$-CIBI condition. Note that we $s \in RS_k(S)$. There are two cases:

- If there is $\mathtt{p}$ such that two receive actions are enabled for $\mathtt{p}$, then they are also enabled at $s$, a contradiction.
- If there is $\mathtt{p}$ such that one receive action is enabled for $\mathtt{p}$, and there is $\xrightarrow{\phi}_k$-path s.t. a conflicting send action is fired, and there is not dependency chain in $\phi$, then we have the situation in $TS_k(S)$, hence we have a contradiction. □

Lemma 36 states that any transition in a given set $L_i$ cannot be disabled by a sequence of transitions not in $L_i$.

**Lemma 36.** *Let $S$ be a system, $s \in RS_k(S)$ s.t. $s$ is $k$-*BI*, and $L_1 \cdots L_n = partition(s)$ (with $n \geqslant 1$). For all $L_i$ (with $1 \leqslant i \leqslant n$) and for all $\phi = \ell_1 \cdots \ell_m$ such that $\forall 1 \leqslant j \leqslant m : \ell_j \notin L_i$, if $s \xrightarrow{\phi}_k s'$, then $\ell \in L_i \implies s' \xrightarrow{\ell}_k$.*

*Proof.* Take $s \in TS_k(S)$, $L_1 \cdot L_n = partition(s)$, $L_i$ $(1 \leqslant i \leqslant n)$, and $\phi$ as defined in the statement. Take any $\ell \in L_i$ and assume there is $s'$ such that $s \xrightarrow{\phi}_k s'$. We show the result by induction on the length of $\phi$ with the additional property that $subj(\ell) \notin \phi$ (note that this implies $q_{\mathtt{p}} = q'_{\mathtt{p}}$).

If $\phi = \epsilon$, then $s = s'$ and we have the result immediately ($s \xrightarrow{\ell}_k$ by Definition 15).

Assume the result holds for $\phi$ and let us show that it holds for $\phi \cdot \ell'$ with $\ell' \notin L_i$. Assume we have $s'$ such that $s \xrightarrow{\phi}_k s' \xrightarrow{\ell'}_k s''$. We have to show that $s'' \xrightarrow{\ell}_k$, knowing that, by induction hypothesis, we have that $s' \xrightarrow{\ell}_k$ and $q_{\mathtt{p}} = q'_{\mathtt{p}}$. There are two cases:

- If $subj(\ell) = subj(\ell')$, then since $s$ is $k$-BI, we have $s \xrightarrow{\ell'}_k$, hence $\ell' \in L_i$, which implies that the premises of this lemma do not hold: a contradiction.
- If $subj(\ell) \neq subj(\ell')$, then we have $q_{\mathtt{p}} = q'_{\mathtt{p}} = q''_{\mathtt{p}}$ and therefore $q''_{\mathtt{p}} \xrightarrow{\ell}$.
  - If $\ell = \mathtt{pq}!a$. The only possibility for $\ell$ to be disabled in $s''$ and enabled in $s'$ is if $|w''_{\mathtt{pq}}| > k$ which is not possible since $subj(\ell') \neq \mathtt{p}$.
  - If $\ell = \mathtt{qp}?a$. The only possibility for $\ell$ to be disabled in $s''$ and enabled in $s'$ is if $w''_{\mathtt{pq}} = \epsilon$ which is not possible since $subj(\ell') \neq \mathtt{p}$. □

**Lemma 37.** *Let $S$ be a system, then for all $s \in RTS_k(S)$ s.t. $s$ is $k$-*BI* and $\ell \in \mathcal{A}$, if $s \xrightarrow{\ell}_k$, then there is $\phi \in \mathcal{A}^*$ such that $s \xrightarrow{\phi}_k \xrightarrow{\ell}_k$ with $subj(\ell) \notin \phi$.*

*Proof.* By assumption that $s \in RTS_k(S)$, $s$ is visited by Algorithm 1.

If $\mathit{partition}(s)$ is invoked on $s$, the fact that $\mathit{subj}(\ell) \notin \phi$ follows from Definition 15, while the fact that $\ell$ is eventually fired follows from the fact that the list of sets of transition decreases at each iteration in Algorithm 1 and Lemma 36.

If $\mathit{partition}(s)$ is not invoked, then we have that $E$ is not empty when $s$ is visited. Let $t$ be a the last node visited before $s$ such that $\mathit{partition}(t)$ is invoked. Pose $L_1 \cdots L_m = \mathit{partition}(t)$ and assume $E = L_i \cdots L_m$ $(i > 1)$ when $s$ is visited. If there is $L_j$ such that $\ell \in L_j$ $(i \leqslant j \leqslant m)$, we have the result as above. Otherwise, there are two cases

- If $\ell$ is independent from all the actions in $L_i \cdots L_m$, then $\ell$ will still be enabled once the list is entirely processed, and therefore $\ell$ will be included in the partition resulting from the next invocation of $\mathit{partition}(\_)$.
- If $\ell$ depends on some partition $L_j$, then we have a contradiction: either $\ell$ is included in $L_j$ (it must have been enabled at $t$) or the list returned by $\mathit{partition}(t)$ is not a partition. □

**Lemma 38.** *Let $S$ be a system. If $s_0 \xrightarrow{\phi_1}_k s \xrightarrow{\ell}_k s' \xrightarrow{\phi_2}_k t$ such that $s$ is $k$-*BI*, $\mathit{subj}(\ell) \notin \phi_2$, $\mathit{chan}(\ell) \notin \phi_2$, and $s \xrightarrow{\ell'}_k$ with $\mathit{subj}(\ell) = \mathit{subj}(\ell')$ then $s \xrightarrow{\ell'}_k s'' \xrightarrow{\phi_2}_k t'$ for some $s''$ and $t'$.*

*Proof.* Assume that $E = L_1 \cdots L_m$ when $s$ is visited by Algorithm 1, then we have $\ell, \ell' \in L_1$ and $s \xrightarrow{\ell'}_k s''$ for some $s''$. When both $s'$ and $s''$ are visited next, we have $E = L_2 \cdots L_m$, hence it is easy to show they have the same behaviour while $E$ is not empty. Say $s_m$ (resp. $s'_m$) is the first state reachable from $s'$ (resp. $s''$) when $E$ is empty. Note that if $\ell$ is a receive action, then we must have $\ell = \ell'$ since $s$ is $k$-BI. Thus, the only differences between $s_m$ and $s'_m$ are:

- the local state of $\mathit{subj}(\ell)$
- the last message of channel $\mathit{chan}(\ell)$

In terms of enabled transition, this means that for all $\hat{\ell}$ such that $\mathit{subj}(\hat{\ell}) \neq \mathit{subj}(\ell)$ and $\mathit{chan}(\hat{\ell}) \neq \mathit{chan}(\ell)$ is enabled at both $s_m$ and $s'_m$. Hence, posing

$$L'_1 \cdots L'_j = \mathit{partition}(s_m) \qquad \text{and} \qquad L''_1 \cdots L''_l = \mathit{partition}(s'_m)$$

and assuming that the position of the partition of $\mathit{subj}(\ell)$ is $i$ (with $1 \leqslant i \leqslant j$ and $i \leqslant l$), it must be the case that all paths of length less than $i$ and not involving $\mathit{chan}(\ell)$ nor $\mathit{subj}(\ell)$ are the same from both $s_m$ and $s'_m$. Instead, any path longer than $i$ must use an action whose subject is $\mathit{subj}(\ell)$ at position $i$, hence does not satisfy the premises of this lemma. □

**Lemma 39.** *Let $S$ be a* reduced $k$-BI *system such that $TS_k(S) = (N, s_0, \Delta)$, $RTS_k(S) = (\hat{N}, s_0, \hat{\Delta})$, and $t_0 \in N \cap \hat{N}$. The following holds:*

1. *If $t_0 \xrightarrow{\phi}_k s$, then $t_0 \xrightarrow{\phi}_k s$, for some $s$.*

*2. If $t_0 \xrightarrow{\phi}_k s$, then there is $\psi$ and $\phi'$ such that $t_0 \xrightarrow{\psi}_k t$ and $s \xrightarrow{\phi'}_k t$ and $\phi \cdot \phi' \asymp \psi$, for some $t$.*

*Proof.* Item (1) follows trivially from Definition 15 and Algorithm 1, since only transitions that exist in $TS_k(S)$ are copied in $RTS_k(S)$.

We show Item (2) by induction on the length of $\phi$. If $\phi = \epsilon$, then we have the result with $\phi' = \psi = \epsilon$. Assume the result holds for $\phi$ and let us show that it holds for $\phi \cdot \ell$. We have the following situation, where the dotted arrows represent executions in $RTS_k(S)$ and $t$ is in $RTS_k(S)$.[4]

$$\begin{tikzcd}
t_0 \arrow[r, "\phi"] \arrow[dr, dotted, bend right, "\psi"'] & s \arrow[r, "\ell"] \arrow[d, "\phi'"] & s' \\
 & t &
\end{tikzcd}$$

Next, we show that there are $t'$, $\hat{s}$, $s''$, and $\psi'$ such that we have:

$$\begin{tikzcd}
s \arrow[r, "\ell"] \arrow[d, "\phi'"'] & s' \arrow[d, "\phi'"] \\
t \arrow[r, "\ell"] \arrow[d, dotted, "\psi'"'] & \hat{s} \arrow[d, "\psi'"] \\
t' \arrow[r, dotted, "\ell"] & s''
\end{tikzcd} \qquad \text{with } \phi' \cdot \psi' \cdot \ell \asymp \ell \cdot \phi' \cdot \psi'$$

We show this by induction on the length of $\phi'$. If $\phi' = \epsilon$, then we have $s = t$ and $s' = \hat{s}$. There are two cases:

- $E = []$ when $t$ is visited by Algorithm 1. In this case, the algorithm continues with $E = L_1 \cdots L_m = \mathit{partition}(s)$, and by Definition 15 there must be $1 \leqslant i \leqslant m$ such that $\ell \in L_i$ (since $\ell$ is enabled at $t$). Since $\ell$ is independent with all $\ell_j$ such that $1 \leqslant j < i$, we have:

$$s = t \xrightarrow{\ell_1 \cdots \ell_{i-1}}_k t' \xrightarrow{\ell}_k s'' \quad \text{and} \quad s = t \xrightarrow{\ell}_k s' = \hat{s} \xrightarrow{\ell_1 \cdots \ell_{i-1}}_k s''$$

  We have the required result with $\psi' = \ell_1 \cdots \ell_{i-1}$.
- $E = L_i \cdots L_m$ $(i > 0)$ when $t$ is visited by Algorithm 1. Then we have two cases:
  - There is $i \leqslant j \leqslant m$ such that $\ell \in L_j$ and we reason as in the case where $E == []$ (but starting at $i$ instead of 1).
  - If $\ell \notin \bigcup_{i \leqslant j \leqslant m} L_j$, then $\ell$ was not enabled when $\mathit{partition}(\hat{t})$ was invoked (for $\hat{t}$ a node visited on the path to $s$). Hence, $\ell$ is independent with

[4] Note that executions in $RTS_k(S)$ are also in $TS_k(S)$ by Item (1).

all actions in $\bigcup_{i\leqslant j\leqslant m} L_j$ and for all $t''$ such that $t \xrightarrow{\ell_i\cdots\ell_m}_k t''$ with $\forall i \leqslant j \leqslant m : \ell_j \in L_j$, we have $t'' \xrightarrow{\ell}_k$. Pose $L'_1 \cdots L'_n = \mathit{partition}(t'')$, then we have that there is $1 \leqslant j \leqslant n$ such that $\ell \in L'_j$. Reasoning as above, we have

$$s = t \xrightarrow{\ell_i\cdots\ell_m}_k t'' \xrightarrow{\ell'_1\cdots\ell'_{j-1}}_k t' \xrightarrow{\ell}_k s''$$

and

$$s = t \xrightarrow{\ell}_k s' \xrightarrow{\ell_i\cdots\ell_m}_k \xrightarrow{\ell'_1\cdots\ell'_{j-1}}_k s''$$

We have the required result with $\psi' = \ell_i \cdots \ell_m \cdot \ell'_1 \cdots \ell'_{j-1}$.

Now, assuming the inner induction hypothesis holds, let us show the result for $\phi' \cdot \ell'$. We have the following situation, where the red parts are what is to be shown:

$$\begin{tikzcd}
s \arrow[r, "\ell"] \arrow[d, "\phi'"'] & s' \arrow[d, "\phi'"] \\
t_i \arrow[r, "\ell"] \arrow[d, "\ell'"'] & s_i \arrow[d, "\ell'"] \\
t \arrow[r, "\ell"] \arrow[d, "\psi'"] & \hat{s} \arrow[d, "\psi'"] \\
t' \arrow[r, "\ell"] & s''
\end{tikzcd}$$

There are two cases.

- If $\mathit{subj}(\ell) \neq \mathit{subj}(\ell')$, then the two actions commute from $t_i$ and we have the result with $\psi' = \epsilon$.
- If $\mathit{subj}(\ell) = \mathit{subj}(\ell')$, then there are two cases:
  - If $\ell = \ell'$, then $t = s_i$ (by determinism) and we have the result with $\phi' = \epsilon$.
  - If $\ell \neq \ell'$, then we must have $\psi = \psi_1 \cdot \ell' \cdot \psi_2$ with $\mathit{subj}(\ell') \notin \psi_2$ (since $\psi \asymp \phi \cdot \phi' \cdot \ell$ by (outer) induction hypothesis). Since $\ell'$ and $\ell$ have the same subject, there is $\hat{t} \in \mathit{RTS}_k(S)$ such that $t_0 \xrightarrow{\psi_1}_k \hat{t}$ such that $\hat{t} \xrightarrow{\ell}_k$ and $\hat{t} \xrightarrow{\ell'}_k$ by $k$-BI.

    Thus, by Lemma 38, we also have $t_0 \xrightarrow{\psi_1}_k \hat{t} \xrightarrow{\ell}_k \xrightarrow{\psi_2}_k t''$ for some $t''$. By (outer) induction hypothesis, we have $\psi = \psi_1 \cdot \ell' \cdot \psi_2 \asymp \phi \cdot \phi' \cdot \ell'$ and since $\mathit{subj}(\ell') \notin \psi_2$, we also have $\psi_1 \cdot \psi_2 \asymp \phi \cdot \phi'$ and $\psi_1 \cdot \ell \cdot \psi_2 \asymp \phi \cdot \phi' \cdot \ell$ hence $s_i = t''$. Since $t''$ is in $\mathit{RTS}_k(S)$, we have the required result with $\psi' = \epsilon$.

Going back to the outer induction, we have to show that

$$\psi \cdot \psi' \cdot \ell \asymp \phi \cdot \ell \cdot \phi''$$

In other words, $\phi \cdot \ell \in TS_k(S)$ can be extended with $\phi''$ so that there is an equivalent execution in $RTS_k(S)$, i.e., $\psi \cdot \psi' \cdot \ell$. By induction hypothesis, we have $\psi \asymp \phi \cdot \phi'$, hence we have

$$\psi \cdot \psi' \cdot \ell \asymp \phi \cdot \phi' \cdot \psi' \cdot \ell$$

From the inner induction, we know that $\phi' \cdot \psi' \cdot \ell \asymp \ell \cdot \psi''$, hence, we have

$$\phi \cdot \phi' \cdot \psi' \cdot \ell \asymp \phi \cdot \ell \cdot \phi''$$

and thus we have the required result. □

**Lemma 40.** *Let $S$ be reduced $k$-*BI*, for all $s \in RS_k(S)$, there is $t \in RTS_k(S)$ such that $s \xrightarrow{\phi}_k t$.*

*Proof.* Since $s \in RS_k(S)$, there is $\psi$ such that $s_0 \xrightarrow{\psi}_k s$. Since $s_0 \in RTS_k(S)$, we can apply Lemma 39 and obtain the required result. □

**Lemma 41.** *If $S$ is reduced $k$-*OBI *and reduced $k$-*SIBI*, then $S$ is $k$-*SIBI*.*

*Proof.* By contradiction. Take $s_0 \xrightarrow{\phi}_k s = (\boldsymbol{q}; \boldsymbol{w}) \in RS_k(S)$.

- If $s \xrightarrow{\mathtt{pr}?a}_k s_1$ and $s \xrightarrow{\mathtt{sr}?b}_k s_2$. Then, by Lemma 39, there is $t \in \hat{N}$ s.t. $s_o \xrightarrow{\psi}_k t$ and $s_1 \xrightarrow{\phi''}_k t$ and $\phi \cdot \mathtt{pr}?a \cdot \phi'' \asymp \psi$. Then both $\mathtt{pr}?a$ and $\mathtt{sr}!b$ must appear in $\psi$, which contradicts the assumption that $S$ is reduced $k$-SIBI.
- If $s \xrightarrow{\mathtt{pr}?a}_k s_1$ and there is $(q_{\mathtt{r}}, \mathtt{sr}?b, q'_{\mathtt{r}}) \in \delta_{\mathtt{r}}$ s.t. $s \xrightarrow{\phi'}_k \xrightarrow{\mathtt{sp}!b}_k s'$. Then we have a contradiction with the assumption that $S$ is reduced $k$-SIBI, via by Lemma 39 as above, with $\phi \cdot \mathtt{pr}?a \cdot \phi' \cdot \mathtt{sp}!b \cdot \phi'' \asymp \psi$. □

**Lemma 42.** *If $S$ is reduced $k$-*OBI *and reduced $k$-*CIBI*, then $S$ is $k$-*CIBI*.*

*Proof.* By contradiction. Take $s_0 \xrightarrow{\phi}_k s = (\boldsymbol{q}; \boldsymbol{w}) \in RS_k(S)$.

- If $s \xrightarrow{\mathtt{pr}?a}_k s_1$ and $s \xrightarrow{\mathtt{sr}?b}_k s_2$. Then, by Lemma 39, there is $t \in \hat{N}$ s.t. $s_o \xrightarrow{\psi}_k t$ and $s_1 \xrightarrow{\phi''}_k t$ and $\phi \cdot \mathtt{pr}?a \cdot \phi'' \asymp \phi$ and $\psi$. Clearly, we must have both $\mathtt{pr}!a$ and $\mathtt{sr}!b$ in $\psi$.
  - If we have

$$\psi = \psi_1 \cdot \mathtt{pr}!a \cdot \psi_2 \cdot \mathtt{sr}!b \cdot \psi_3 \cdot \mathtt{pr}?a \cdot \psi_4, \text{or}$$
$$\psi = \psi_1 \cdot \mathtt{sr}!b \cdot \psi_2 \cdot \mathtt{pr}!a \cdot \psi_3 \cdot \mathtt{pr}?a \cdot \psi_4$$

    where $\psi_1$, $\psi_2$, and $\psi_3$ have been chosen appropriately so that the send actions are one matched at $s$, then we have a contradiction with the assumption that $S$ is $k$-CIBI (both messages can be consumed).

  - Assume we have
$$\psi = \psi_1 \cdot \mathtt{pr}!a \cdot \psi_2 \cdot \mathtt{pr}?a \cdot \psi_3 \cdot \mathtt{sr}!b \cdot \psi_4$$
where $\psi_1$, $\psi_2$, and $\psi_3$ have been chosen appropriately so that the send actions are one matched at $s$. Since $S$ is reduced $k$-CIBI, we must have $\hat{s} \vdash \mathtt{pr}?a \prec_{\psi_3} \mathtt{sr}!b$, with $\hat{s}$ such that $s_0 \xrightarrow{\psi_1 \cdot \mathtt{pr}!a \cdot \psi_2}_k \hat{s}$. However, $\mathtt{pr}!a$ and $\mathtt{sr}!b$ appear in $\phi$, which contradicts the existence of a dependency chain between $\mathtt{pr}?a$ and $\mathtt{sr}!b$ by Lemma 25.

- If $s \xrightarrow{\mathtt{pr}?a}_k s_1$ and there is $(q_\mathtt{r}, \mathtt{sr}?b, q'_\mathtt{r}) \in \delta_\mathtt{r}$ s.t. $s_1 \xrightarrow{\phi'}_k \xrightarrow{\mathtt{sp}!b}_k s'$ with $\neg(s \vdash \mathtt{pr}?a \prec_{\phi'} \mathtt{sr}!b)$. Then, by Lemma 39, there is $t \in \hat{N}$ s.t. $s_0 \xrightarrow{\psi}_k t$, $s_1 \xrightarrow{\phi''}_k t$, and
$$\phi \cdot \mathtt{pr}?a \cdot \phi' \cdot \mathtt{sp}!b \cdot \phi'' \asymp \psi$$
There are two cases depending on the structure of $\psi$:
  - If $\mathtt{sp}!b$ appears before $\mathtt{pr}?a$ in $\psi$, then we have a contradiction with the assumption that $S$ is reduced $k$-CIBI.
  - If $\mathtt{sp}!b$ appears after $\mathtt{pr}?a$, then pose
$$\psi = \psi_1 \cdot \mathtt{pr}?a \cdot \psi_2 \cdot \mathtt{sp}!b \cdot \psi_3$$
Since $S$ is reduced $k$-CIBI, we must have $\hat{s} \vdash \mathtt{pr}?a \prec_{\psi_2} \mathtt{sp}!b$ assuming $\hat{s}$ is such that $s_0 \xrightarrow{\psi_1}_k \hat{s}$. By Lemma 25, we have a contradiction with the assumption that $\neg(s \vdash \mathtt{pr}?a \prec_{\phi'} \mathtt{sr}!b)$. □

**Theorem 11.** *Let $S$ be reduced $k$-*OBI*. $S$ is reduced $k$-*SIBI *iff $S$ is $k$-*SIBI*.*

*Proof.* By Lemma 41 and Lemma 34. □

**Lemma 43.** *Let $S$ be reduced $k$-*BI*, if $S$ is $k$-exhaustive, then $S$ is also* reduced *$k$-exhaustive.*

*Proof.* We show that Definition 9 applies to every state $s \in RTS_k(S) \subseteq TS_k(S)$. By assumption, we have that for every $\mathtt{p} \in \mathcal{P}$, if $q_\mathtt{p}$ is a sending state, then $\forall (q_\mathtt{p}, \ell, q'_\mathtt{p}) \in \delta_\mathtt{p} : \exists \phi \in \mathcal{A}^* : s \xrightarrow{\phi}_k \xrightarrow{\ell}_k$ and $\mathtt{p} \notin \phi$. By Lemma 39, there is $\phi'$ and $\psi$ such that $s \xrightarrow{\psi}_k$ and $\phi \cdot \ell \cdot \phi' \asymp \psi$. This implies that we have $\psi = \psi_1 \cdot \ell \cdot \psi_2$ with $subj(\ell) \notin \psi_1$, and $s \xrightarrow{\psi_1 \cdot \ell}_k$, the required result. □

**Lemma 44.** *Let $S$ be reduced $k$-*BI*, if $S$ is* reduced *$k$-exhaustive, then $S$ is also $k$-exhaustive.*

*Proof.* By contradiction, take $s \in TS_k(S)$ such that the $k$-exhaustivity property does not hold (i.e., there is $\mathtt{pq}!a$ that cannot be fired within bound $k$). By Lemma 40, there is $t \in RTS_k(S)$ and $\phi$ such that $s \xrightarrow{\phi}_k t$. Then either $\mathtt{pq}!a$ is in $\phi$, i.e., we have a contradiction, or $\mathtt{p}$ is in the same state in $t$. By assumption, there is $\psi$ such that $t \xrightarrow{\psi \cdot \mathtt{pq}!a}_k$, and by Lemma 39 we also have $t \xrightarrow{\psi \cdot \mathtt{pq}!a}_k$, a contradiction. □

**Theorem 12.** *Let $S$ be reduced $k$-*BI*, $S$ is* reduced *$k$-exhaustive iff $S$ is $k$-exhaustive.*

*Proof.* By Lemma 43 and Lemma 44. □

**Theorem 6.** *Let $S$ be reduced $k$-*OBI *and reduced $k$-*IBI*. (1) $S$ is* reduced *$k$-safe iff $S$ is $k$-safe. (2) $S$ is* reduced *$k$-exhaustive iff $S$ is $k$-exhaustive.*

*Proof.* By Theorem 13 and Theorem 12 □

**Lemma 45.** *Let $S$ be reduced $k$-*BI*, if $S$ is $k$-safe, then $S$ is also* reduced *$k$-safe.*

*Proof.* The proof works similarly to the proof of Lemma 43. We show that Definition 4 applies to every state in $s \in RTS_k(S) \subseteq TS_k(S)$. Each condition follows easily by showing the existence of an equivalent execution, by Lemma 39. □

**Lemma 46.** *Let $S$ be reduced $k$-*BI*, if $S$ is* reduced *$k$-safe, then $S$ is also $k$-safe.*

*Proof.* The proof works similarly to the proof of Lemma 44. By contradiction, we assume that there is a state $s$ for which the properties of Definition 4 do not hold. Using Lemma 40, we show that there is an execution from $s$ to a state in $RTS_k(S)$ for which the properties hold by assumption. □

**Theorem 13.** *Let $S$ be reduced $k$-*BI*, $S$ is* reduced *$k$-safe iff $S$ is $k$-safe.*

*Proof.* By Lemma 45 and Lemma 46. □

**Lemma 4.** *Let $S$ be a system such that $RTS_k(S) = (\hat{N}, s_0, \hat{\Delta})$, for all $\phi$ and $\phi'$ such that $s_0 \xrightarrow{\phi}_k$ and $s_0 \xrightarrow{\phi'}_k$, we have that: $\phi \asymp \phi' \implies \phi = \phi'$.*

*Proof.* We show that $\phi \neq \phi' \implies \neg(\phi \asymp \phi')$. Let $\psi$ be the longest common prefix of $\phi$ and $\phi'$. Take $s$ such that $s_0 \xrightarrow{\psi}_k s$. Since $\phi \neq \phi'$, we must have $\ell$ and $\ell'$ such that $s \xrightarrow{\ell}_k$ and $s \xrightarrow{\ell'}_k$. However, since $\phi \asymp \phi'$, it must be the case that $subj(\ell) \neq subj(\ell')$; which gives us a contradiction since we have that $s \xrightarrow{\ell}_k$ and $s \xrightarrow{\ell'}_k$, while $\ell$ and $\ell'$ must be in different sets $L_i$ and $L_j$. □

**Theorem 14.** *Let $S$ be reduced $k$-*OBI*. $S$ is reduced $k$-*CIBI *iff $S$ is $k$-*CIBI*.*

*Proof.* By Lemma 35 and 42. □

**Theorem 4.** *Let $S$ be reduced $k$-*OBI*. $S$ is reduced $k$-*CIBI *(resp. $k$-*SIBI*) iff $S$ is $k$-*CIBI *(resp. $k$-*SIBI*).*

*Proof.* By Theorem 11 and Theorem 14. □

# F Proofs for Section 5

**Lemma 47.** *Let $S$ be a system. If $s_0 \xrightarrow{\phi}_k$, then $\phi$ is $k$-match-bounded.*

*Proof.* We first note that $\phi$ is valid, by Lemma 13. We have to show that for any prefix $\psi$ of $\phi$, we have

$$min\{|\pi^!_{\mathtt{pq}}(\psi)|, |\pi^?_{\mathtt{pq}}(\phi)|\} - |\pi^?_{\mathtt{pq}}(\psi)| \leqslant k$$

There are two cases:

- If $|\pi^!_{\mathtt{pq}}(\psi)| \leqslant |\pi^?_{\mathtt{pq}}(\phi)|$, we have the result immediately since
$$|\pi^!_{\mathtt{pq}}(\psi)| - |\pi^?_{\mathtt{pq}}(\psi)| \leqslant k$$
by hypothesis (and the definition of $k$-boundedness).
- If $|\pi^!_{\mathtt{pq}}(\psi)| > |\pi^?_{\mathtt{pq}}(\phi)|$ then the following holds
$$|\pi^?_{\mathtt{pq}}(\phi)| - |\pi^?_{\mathtt{pq}}(\psi)| < |\pi^!_{\mathtt{pq}}(\psi)| - |\pi^?_{\mathtt{pq}}(\psi)| \leqslant k$$
by hypothesis, and we have the required result. □

## F.1 Proofs for Section 5.1

**Lemma 48.** *If $\phi \cdot \ell \cdot \phi' \in \mathcal{A}^*$ is a valid $k$-match-bounded execution such that $subj(\ell) \notin \phi'$ and $\phi \cdot \phi'$ is also valid, then $\phi \cdot \phi'$ is a $k$-match-bounded execution.*

*Proof.* We note that we only have to consider the number of messages on the channel of $\ell$, as the others are unchanged. There are two cases depending on the direction of $\ell$.

- If $\ell = \mathtt{pq}!a$, then the result follows trivially since the number of send actions strictly decreases.
- If $\ell = \mathtt{pq}?a$, we separate the prefixes of $\phi \cdot \ell \cdot \phi'$ depending on whether they include $\ell$ or not.
  1. For each prefix $\psi$ of $\phi$, we have
  $$min\{|\pi^!_{\mathtt{pq}}(\psi)|, |\pi^?_{\mathtt{pq}}(\phi \cdot \ell \cdot \phi')|\} - |\pi^?_{\mathtt{pq}}(\psi)| \leqslant k$$
  by hypothesis. We have to show that
  $$min\{|\pi^!_{\mathtt{pq}}(\psi)|, |\pi^?_{\mathtt{pq}}(\phi \cdot \phi')|\} - |\pi^?_{\mathtt{pq}}(\psi)| \leqslant k$$
  which follows trivially since $|\pi^?_{\mathtt{pq}}(\phi \cdot \phi')| = |\pi^?_{\mathtt{pq}}(\phi \cdot \ell \cdot \phi')| - 1$.
  2. For each prefix of $\psi$ of $\phi'$, we have to show that
  $$min\{|\pi^!_{\mathtt{pq}}(\phi \cdot \psi)|, |\pi^?_{\mathtt{pq}}(\phi \cdot \phi')|\} - |\pi^?_{\mathtt{pq}}(\phi \cdot \psi)| \leqslant k$$
  By hypothesis ($subj(\ell) \notin \phi'$), we have $|\pi^?_{\mathtt{pq}}(\phi')| = 0$ and since $\phi \cdot \ell \cdot \phi'$ is valid by assumption, we have $|\pi^!_{\mathtt{pq}}(\phi)| \geqslant |\pi^?_{\mathtt{pq}}(\phi)|$, hence we are left to show that
  $$|\pi^?_{\mathtt{pq}}(\phi)| - |\pi^?_{\mathtt{pq}}(\phi \cdot \psi)| \leqslant k$$
  Similarly, we know that $|\pi^!_{\mathtt{pq}}(\phi \cdot \ell)| - |\pi^?_{\mathtt{pq}}(\phi \cdot \ell \cdot \psi)| \leqslant k$. We have the result since $|\pi^?_{\mathtt{pq}}(\phi)| = |\pi^?_{\mathtt{pq}}(\phi \cdot \ell)| - 1$ and $|\pi^?_{\mathtt{pq}}(\phi \cdot \psi)| = |\pi^?_{\mathtt{pq}}(\phi \cdot \ell \cdot \psi)| - 1$. □

**Lemma 49.** *If $S$ is (reduced) $k$-OBI, IBI, and $k$-exhaustive system, then it is existentially $k$-bounded.*

*Proof.* Take $s$ and $\phi$ such that $s_0 \xrightarrow{\phi} s$. By Lemma 19, there is $t \in RS_k(S)$, $\phi'$ and $\psi$ such that $s \xrightarrow{\phi'} t$, $s_0 \xrightarrow{\psi}_k t$, and $\phi \cdot \phi' \Bumpeq \psi$. Note that $\psi$ is valid by Lemma 13 and $k$-match-bounded by Lemma 47. We show that there is a $k$-match-bounded execution that leads to $s$ by inductively deconstructing $\phi'$, starting from its last element. **(Base case)** If $\phi' = \epsilon$, then we have the results immediately by Lemma 19, i.e., we have $\phi \cdot \epsilon \Bumpeq \psi$ with $\psi$ $k$-match-bounded.
**(Inductive case)** Take $\phi' = \phi_1 \cdot \ell$. From Lemma 19, there is $\psi$ ($k$-bounded) such that $\phi \cdot \phi_1 \cdot \ell \Bumpeq \psi$. Since the two executions are $\Bumpeq$-equivalent, we must have $\psi = \psi_0 \cdot \ell \cdot \psi_1$ with $subj(\ell) \notin \psi_1$. Hence, we have the following situation, where the dashed execution is due to the fact that $subj(\ell) \notin \psi_1$ (i.e., $\ell$ is independent from $\psi_1$):

$$\begin{tikzcd}
s_0 \arrow[r, "\phi"] \arrow[d, "\psi_0"'] & s \arrow[d, "\phi_1"] \\
s' \arrow[r, dashed, "\psi_1"] \arrow[d, "\ell"'] & t \arrow[d, "\ell"] \\
s'' \arrow[r, "\psi_1"] & t'
\end{tikzcd}$$

where $\psi_0 \cdot \psi_1$ is valid by Lemma 13, and $k$-match-bounded by Lemma 48. Next, we repeat the procedure posing $\psi := \psi_0 \cdot \psi_1$ and $\phi' := \phi_1$. We note that the procedure always terminates since the execution $\phi'$ strictly decrease at each iteration. □

**Lemma 50.** *If $\phi_0 \cdot \phi_1$ is $k$-match-bounded and*

$$\forall \mathsf{pq} \in \mathcal{C} : |\pi^{!}_{\mathsf{pq}}(\phi_0)| \leqslant |\pi^{?}_{\mathsf{pq}}(\phi_0 \cdot \phi_1)|$$

*then $\phi_0$ is $k$-bounded for $s_0$.*

*Proof.* Pick any $\mathsf{pq} \in \mathcal{C}$. By definition of $k$-match-bounded, for each prefix $\psi$ of $\phi_0 \cdot \phi_1$, we have:

$$min\{|\pi^{!}_{\mathsf{pq}}(\psi)|, |\pi^{?}_{\mathsf{pq}}(\phi_0 \cdot \phi_1)|\} - |\pi^{?}_{\mathsf{pq}}(\psi)| \leqslant k$$

In particular, for each prefix $\psi_0$ of $\phi_0$, we have $min\{|\pi^{!}_{\mathsf{pq}}(\psi_0)|, |\pi^{?}_{\mathsf{pq}}(\phi_0 \cdot \phi_1)|\} - |\pi^{?}_{\mathsf{pq}}(\psi_0)| \leqslant k$. By assumption and the fact that $\psi_0$ is a prefix of $\phi_0$, we have

$$|\pi^{!}_{\mathsf{pq}}(\psi_0)| \leqslant |\pi^{!}_{\mathsf{pq}}(\phi_0)| \leqslant |\pi^{?}_{\mathsf{pq}}(\phi_0 \cdot \phi_1)|$$

Hence, $min\{|\pi^{!}_{\mathsf{pq}}(\psi_0)|, |\pi^{?}_{\mathsf{pq}}(\phi_0 \cdot \phi_1)|\} = |\pi^{!}_{\mathsf{pq}}(\psi_0)|$ and $|\pi^{!}_{\mathsf{pq}}(\psi_0)| - |\pi^{?}_{\mathsf{pq}}(\psi_0)| \leqslant k$, as required. □

**Lemma 51.** *If $S$ is $\exists$-$k$-bounded and has the eventual reception property, then $S$ is $k$-exhaustive.*

*Proof.* **($k$-eventual reception)** We first show that for all $s = (\boldsymbol{q}; \boldsymbol{w}) \in RS_k(S)$, if $w_{\mathtt{pq}} = a \cdot w$, then $s \rightarrow_k{}^* \xrightarrow{\mathtt{pq}!a}_k$. Take $\phi_0$ such that $s_0 \xrightarrow{\phi_0}_k s$. By eventual reception, we have that $s \xrightarrow{\phi_1}\xrightarrow{\mathtt{pq}?a} t$, for some $\phi_1$ and $t$. Take $\phi_2$ such that $t \xrightarrow{\phi_2}$ and

$$\forall \mathtt{pq} \in \mathcal{C} : |\pi^!_{\mathtt{pq}}(\phi_0 \cdot \phi_1)| \leqslant |\pi^?_{\mathtt{pq}}(\phi_0 \cdot \phi_1 \cdot \mathtt{pq}?a \cdot \phi_2)|$$

there is such $\phi_2$ by the eventual reception property. Since $S$ is existentially bounded, there is $\psi$ such that $\psi$ is $k$-match-bounded and $\psi \Bumpeq \phi_0 \cdot \phi_1 \cdot \mathtt{pq}?a \cdot \phi_2$.

Next, remove all actions in $\phi_0$ from $\psi$ as follows. Take the first action in $\phi_0$ (i.e., a send action) and remove it from $\psi$ as well as its receive counterpart, if any. If this action is not received within $\phi_0$, then store it in $\hat{\psi}$. Repeat until all actions from $\phi_0$ have been removed, so to obtain the sequence: $\hat{\psi} \cdot \psi_1$ which is $k$-match-bounded and valid, so that we have $\hat{\psi} \cdot \psi_1 \Bumpeq \hat{\psi} \cdot \phi_1 \cdot \mathtt{pq}?a \cdot \phi_2$.

Pose $\psi_1 = \psi_2 \cdot \mathtt{pq}?a \cdot \psi_3$ and let us show that $\psi_2 \cdot \mathtt{pq}?a$ is $k$-bounded for $s$, by showing that $\hat{\psi} \cdot \psi_2 \cdot \mathtt{pq}?a$ is $k$-bounded. We have to show that all prefixes are $k$-bounded. This is trivial for any prefix of $\hat{\psi}$ since $s \in RS_k(S)$. For any prefix $\hat{\psi}_2$ of $\psi_2$ we have to show that

$$\forall \mathtt{pq} \in \mathcal{C} : |\pi^!_{\mathtt{pq}}(\hat{\psi} \cdot \hat{\psi}_2)| - |\pi^?_{\mathtt{pq}}(\hat{\psi} \cdot \hat{\psi}_2)| \leqslant k$$

Since $\hat{\psi} \cdot \psi_1$ is $k$-match-bounded, we have

$$\forall \mathtt{pq} \in \mathcal{C} : min\{|\pi^!_{\mathtt{pq}}(\hat{\psi} \cdot \hat{\psi}_2)|, |\pi^?_{\mathtt{pq}}(\hat{\psi} \cdot \psi_2 \cdot \mathtt{pq}?a \cdot \psi_3)|\} - |\pi^?_{\mathtt{pq}}(\hat{\psi} \cdot \hat{\psi}_2)| \leqslant k$$

By construction, we have $|\pi^!_{\mathtt{pq}}(\hat{\psi} \cdot \hat{\psi}_2)| \leqslant |\pi^?_{\mathtt{pq}}(\hat{\psi} \cdot \psi_2 \cdot \mathtt{pq}?a \cdot \psi_3)|$, hence we have the required result.
**($k$-exhaustivity)** We show the rest by contradiction. Assume there is $s \in RS_k(S)$ for which the $k$-exhaustivity condition does not hold. Hence, there must be $\mathtt{pq} \in \mathcal{C}$ such that $|w_{\mathtt{pq}}| = k \geqslant 1$. From the result above, we have $s \rightarrow_k{}^* \xrightarrow{\mathtt{pq}?a}_k t$ for some $a$, and therefore we have $t \xrightarrow{\mathtt{pq}!b}_k$, for any $b$, a contradiction. □

**Lemma 52.** *If $S$ is existentially $k$-bounded and safe, then for any $k$-match-bounded $\phi$ such that $s_0 \xrightarrow{\phi} s$, there are $\psi$ and $\phi'$ such that $s_0 \xrightarrow{\psi}_k t$ and $s \xrightarrow{\phi'} t$ and $\psi \Bumpeq \phi \cdot \phi'$.*

*Proof.* Take $\phi$ $k$-match-bounded s.t. $s_0 \xrightarrow{\phi} s$. By safety, there is $\phi'$ such that $s \xrightarrow{\phi'}$ with $\forall \mathtt{pq} \in \mathcal{C} : |\pi^!_{\mathtt{pq}}(\phi)| \leqslant |\pi^?_{\mathtt{pq}}(\phi \cdot \phi')|$, i.e., we extend $\phi$ with an execution that consumes all messages sent in $\phi$.

Since $S$ is existentially bounded, there is $\psi \in [\phi \cdot \phi']_{\Bumpeq} \cap \mathcal{A}^*|_k$. Take prefix $\psi_0$ of $\psi$ such that $\exists \phi'' : \forall \mathtt{p} \in \mathcal{P} : \pi_{\mathtt{p}}(\psi_0) = \pi_{\mathtt{p}}(\phi \cdot \phi'')$. If $\psi_0$ is $k$-bounded, we have the required result, otherwise, there must be a prefix $\psi_1$ of $\psi_0$ such that

$$|\pi^!_{\mathtt{pq}}(\psi_1)| - |\pi^?_{\mathtt{pq}}(\psi_1)| > k$$

However, since $\psi$ is $k$-match-bounded, we have

$$min\{|\pi^{!}_{\mathtt{pq}}(\psi_1)|, |\pi^{?}_{\mathtt{pq}}(\psi)|\} - |\pi^{?}_{\mathtt{pq}}(\psi_1)| \leqslant k$$

and by construction of $\psi \Bumpeq \phi \cdot \phi'$, we have $|\pi^{!}_{\mathtt{pq}}(\psi_1) \leqslant |\pi^{?}_{\mathtt{pq}}(\psi)|$, i.e., a contradiction. □

**Theorem 7.** *(1) If $S$ is (reduced) $k$-*OBI, IBI*, and $k$-exhaustive, then it is existentially $k$-bounded. (2) If $S$ is existentially $k$-bounded and has the eventual reception property, then it is $k$-exhaustive.*

*Proof.* Part (1) follows from Lemma 49 and Part (2) follows from Lemmas 51. □

### F.2 Proofs for Section 5.2

**Lemma 5.** *Let $S$ be a system and $\phi \in \mathcal{A}^*$ such that $s_0 \xrightarrow{\phi} s = (\boldsymbol{q}; \boldsymbol{\epsilon})$, then $\phi$ is $k$-match-bounded if and only if $\phi$ is $k$-bounded for $s_0$.*

*Proof.* The ($\Leftarrow$) direction follows from Lemma 47. The ($\Rightarrow$) direction follows from the fact that for any prefix $\psi$ of $\phi$, we have

$$|\pi^{!}_{\mathtt{pq}}(\psi)| \leqslant |\pi^{?}_{\mathtt{pq}}(\phi)|$$

since all messages sent along $\phi$ are received (all channels in $s$ are empty). Hence we have $|\pi^{!}_{\mathtt{pq}}(\psi)| - |\pi^{?}_{\mathtt{pq}}(\psi)| \leqslant k$ by Definition 22, i.e., $\phi$ is $k$-bounded. □

**Theorem 8.** *(1) If $S$ is existentially $k$-bounded, then it is existentially* stable *$k$-bounded. (2) If $S$ is existentially* stable *$k$-bounded and has the stable property, then it is existentially $k$-bounded.*

*Proof.* We show both statements by contradiction.

1. Assume by contradiction that $S$ is existentially $k$-bounded, but *not* existentially stable $k$-bounded. Then, there must be $\phi$ such that $s_0 \xrightarrow{\phi} s = (\boldsymbol{q}; \boldsymbol{\epsilon})$ where $\phi$ has no $\Bumpeq$ equivalent execution which is $k$-bounded for $s_0$. However, since $S$ is existentially $k$-bounded, there is $\psi \Bumpeq \phi$ such that $\psi$ is $k$-match-bounded. Since $s_0 \xrightarrow{\psi} (\boldsymbol{q}; \boldsymbol{\epsilon})$, by Lemma 5, $\psi$ is $k$-bounded, a contradiction.
2. Assume by contradiction that $S$ is existentially stable $k$-bounded and has the stable property, but *not* existentially $k$-bounded. Then there is $\phi$ such that $s_0 \xrightarrow{\phi} s = (\boldsymbol{q}; \boldsymbol{w})$ (with $\boldsymbol{q}$ not empty) such that $\phi$ has no $\Bumpeq$ equivalent execution which is $k$-match-bounded for $s_0$. Since $S$ has the stable property, we have $s \xrightarrow{\phi'}$ and there is $\psi \Bumpeq \phi \cdot \phi'$ such that $\psi$ is $k$-bounded (since $S$ is $\exists$S-$k$-bounded). Then we reason as for the proof of Lemma 49 and progressively deconstruct $\phi'$ to show that there is a subsequence of $\psi$ that is $k$-match-bounded and $\Bumpeq$-equivalent to $\phi$, a contradiction. □

**Lemma 53.** *Let $S$ be $\exists$-$k$-bounded, then for all* stable *configurations $s$ and $s'$ in $RS(S)$ such that $s \xrightarrow{\phi} s'$, there is $\psi \Bumpeq \phi$ such that $\psi$ is $k$-bounded (for $s$).*

*Proof.* Since $s$ is stable and $S$ is $\exists$-$k$-bounded, there is $\phi_0$ $k$-bounded for $s_0$ such that $s_0 \xrightarrow{\phi_0}_k s$, and we have $\hat{\psi}$ $k$-bounded such that $\hat{\psi} \Bumpeq \phi_0 \cdot \phi$. We show that we inductively remove the actions of $\phi_0$ from $\hat{\phi}$ while preserving its $k$-boundedness. Since $s$ and $s'$ are stable, we have $\phi_0 = \mathtt{pq}!a \cdot \phi'_1 \cdot \mathtt{pq}?a \cdot \phi'_2$, with $\pi^{?}_{\mathtt{pq}}(\phi'_1) = \epsilon$. Hence, we can remove the first respective occurrence of $\mathtt{pq}!a$ and $\mathtt{pq}?a$ from $\hat{\psi}$ without affecting its $k$-boundedness: $(i)$ the new execution is still valid since we remove a send and its receive and $(ii)$ the bound is preserved since we remove a send and a receive simultaneously. We repeat the procedure until all the elements of $\phi_0$ have been removed and we obtain the required result. □

**Lemma 6.** *Let $S$ be an existentially stable $k$-bounded system with the stable property, then for all $s \in RS_k(S)$, there is $t$ stable such that $s \rightarrow_k{}^* t$.*

*Proof.* First we observe that for any stable $t$, we have $t \in RS_k(S)$ since $S$ is $\exists$S-$k$-bounded, by Lemma 5. Assume $t_0$ is stable and $t_0 \xrightarrow{\phi}_k s$. We show the result by induction on the length of $\phi$.

If $\phi = \ell$, then we have the result since $t_0$ is stable and there is stable $t'$ such that $t_0 \xrightarrow{\ell}_k s \rightarrow^* t'$ since $S$ has the stable property. Finally, by Lemma 53, we have $s \rightarrow_k{}^* t'$.

Assume the result holds for $\phi$ and let us show that it holds for $\phi \cdot \ell$. Pose $t_0 \xrightarrow{\phi}_k s \xrightarrow{\ell}_k s'$. By induction hypothesis, we have that $s \xrightarrow{\phi'}_k t$ for some $t$ stable and $\phi' \in \mathcal{A}^*$. We have to show that $s' \rightarrow_k{}^* t'$ with $t'$ stable. There are two cases:

- If $\mathit{subj}(\ell) \notin \phi'$, then we have $s' \xrightarrow{\phi} t'$ and $t \xrightarrow{\ell}_k t'$, and we only have to show that $s' \xrightarrow{\phi}_k t'$, which follows trivially from the fact that $\mathit{subj}(\ell) \notin \phi'$ (i.e., there is no other send on the channel in $\phi'$).
- If $\mathit{subj}(\ell) \in \phi'$, then there are two sub-cases depending on the direction of $\ell$.
  - If $\ell$ is a receive action, then the result follows trivially.
  - If $\ell$ is a send action. Assume w.l.o.g. that $\phi' = \phi'_1 \cdot \ell \cdot \phi'_2$ with $\mathit{subj}(\ell) \notin \phi'_1$, then we have $s' \xrightarrow{\phi'_1}_k \xrightarrow{\phi'_2}_k t' = t$, and we have the required result.

We have shown that either there is stable $t$ such that $t \xrightarrow{\ell}_k t'$, hence we are back to the base case, or $t = t'$, in which case the result follows trivially. □

**Theorem 9.** *Let $S$ be an $\exists$(S)-$k$-bounded system with the stable property, then it is $k$-exhaustive.*

*Proof.* We first note that by Theorem 8 we have that $S$ is both $\exists$S-$k$-bounded and $\exists$-$k$-bounded since it has the stable property. Assume by contradiction, that $S$ is not $k$-exhaustive. Then, there is $s$ such that $s_0 \xrightarrow{\phi}_k s = (\boldsymbol{q}; \boldsymbol{w})$ and $\mathtt{p}$ such that $(q_{\mathtt{p}}, \mathtt{pq}!a, q'_{\mathtt{p}}) \in \delta_{\mathtt{p}}$ and $\neg(s \rightarrow_k{}^* \xrightarrow{\mathtt{pq}!a}_k)$. By Lemma 6, there is stable $t$ such that $s \xrightarrow{\psi}_k t$. Then either $\mathtt{p} \in \psi$ and therefore $\mathtt{pq}!a$ can be fired in $\psi$ and we have a contradiction, or $\mathtt{p} \notin \psi$ and $t \xrightarrow{\mathtt{pq}!a}_k$, i.e., another contradiction. □

### F.3 Proofs for Section 5.3

**Lemma 7.** *Let $\phi$ be a valid execution. If $\phi$ is a $k$-exchange then it is a $k$-match-bounded execution.*

*Proof.* Since $\phi$ is a $k$-exchange, it must be of the form

$$\phi = \phi_1 \cdot \psi_1 \cdots \phi_n \cdot \psi_n \quad \text{where } \forall 1 \leqslant i \leqslant n : \phi_i \in \mathcal{A}_!^* \wedge \psi_i \in \mathcal{A}_?^* \wedge |\phi_i| \leqslant k$$

We must show that for every prefix $\hat{\phi}$ of $\phi$ and every $\mathsf{pq} \in \mathcal{C}$, the following holds:

$$min\{|\pi^!_{\mathsf{pq}}(\hat{\phi})|, |\pi^?_{\mathsf{pq}}(\phi)|\} - |\pi^?_{\mathsf{pq}}(\hat{\phi})| \leqslant k$$

We first observe that, for all $1 \leqslant i \leqslant n$, if $\hat{\phi} = \phi_1 \cdot \psi_1 \cdots \phi_i$ is $k$-match-bounded, then so is $\hat{\phi} \cdot \psi_i$ (since $\psi_i \in \mathcal{A}_?^*$), hence we only show the result for the former. Take $\mathsf{pq} \in \mathcal{C}$ and pose $\hat{\phi} = \phi_1 \cdot \psi_1 \cdots \phi_i$ (with $1 \leqslant i \leqslant n$). There are two cases:

– If for all $1 \leqslant j < i : \pi^!_{\mathsf{pq}}(\phi_j) = \pi^?_{\mathsf{pq}}(\psi_j)$, then all messages sent on channel $\mathsf{pq}$ are received within each exchange.
  - Case $|\pi^!_{\mathsf{pq}}(\hat{\phi})| \leqslant |\pi^?_{\mathsf{pq}}(\phi)|$. We have

$$\begin{aligned} |\pi^!_{\mathsf{pq}}(\hat{\phi})| &= |\pi^!_{\mathsf{pq}}(\phi_1 \cdots \phi_i)| \\ &= |\pi^?_{\mathsf{pq}}(\psi_1 \cdots \psi_{i-1})| + |\pi^!_{\mathsf{pq}}(\phi_i)| \\ &= |\pi^?_{\mathsf{pq}}(\hat{\phi})| + |\pi^!_{\mathsf{pq}}(\phi_i)| \end{aligned}$$

    Hence, $|\pi^!_{\mathsf{pq}}(\hat{\phi})| - |\pi^?_{\mathsf{pq}}(\hat{\phi})| = |\pi^!_{\mathsf{pq}}(\phi_i)| \leqslant k$, and we have the required result.
  - Case $|\pi^!_{\mathsf{pq}}(\hat{\phi})| > |\pi^?_{\mathsf{pq}}(\phi)|$. Then, there is $i \leqslant m \leqslant n$ such that $\pi^!_{\mathsf{pq}}(\phi_m) \neq \pi^?_{\mathsf{pq}}(\psi_m)$ and we have

$$\begin{aligned} \pi^?_{\mathsf{pq}}(\phi) &= |\pi^?_{\mathsf{pq}}(\psi_1 \cdots \psi_m)| \\ &\geqslant |\pi^?_{\mathsf{pq}}(\psi_1 \cdots \psi_i)| \\ &= |\pi^!_{\mathsf{pq}}(\phi_1 \cdots \phi_i)| = |\pi^!_{\mathsf{pq}}(\hat{\phi})| \end{aligned}$$

    Hence, we obtain $\pi^?_{\mathsf{pq}}(\phi) \geqslant |\pi^!_{\mathsf{pq}}(\hat{\phi})|$, a contradiction with this case.

– If there is $j < i$ such that $\pi^!_{\mathsf{pq}}(\phi_j) \neq \pi^?_{\mathsf{pq}}(\psi_j)$ (take smallest such $j$), then for all $j < m \leqslant n : \pi^?_{\mathsf{pq}}(\psi_m) = \epsilon$, i.e., all messages sent after $j$ are not matched. Hence, we have

$$|\pi^?_{\mathsf{pq}}(\hat{\phi})| = |\pi^?_{\mathsf{pq}}(\psi_1 \cdots \psi_j)| = |\pi^?_{\mathsf{pq}}(\phi)| \tag{6}$$

  Thus, we have

$$\begin{aligned} \pi^!_{\mathsf{pq}}(\hat{\phi}) &= |\pi^?_{\mathsf{pq}}(\psi_1 \cdots \psi_{j-1})| + |\pi^!_{\mathsf{pq}}(\phi_j)| + |\pi^!_{\mathsf{pq}}(\phi_{j+1} \cdots \phi_i)| \\ &\geqslant |\pi^?_{\mathsf{pq}}(\psi_1 \cdots \psi_{j-1})| + |\pi^?_{\mathsf{pq}}(\psi_j)| + |\pi^!_{\mathsf{pq}}(\phi_{j+1} \cdots \phi_i)| \\ &\geqslant |\pi^?_{\mathsf{pq}}(\psi_1 \cdots \psi_j)| = |\pi^?_{\mathsf{pq}}(\phi)| \end{aligned}$$

  Hence, we only have to show that $|\pi^?_{\mathsf{pq}}(\phi)| - |\pi^?_{\mathsf{pq}}(\hat{\phi})| \leqslant k$, which holds by (F.3). □

**Theorem 10.** *(1) If $S$ is $k$-synchronisable, then it is existentially $k$-bounded. (2) If $S$ is $k$-synchronisable and has the eventual reception property, then it is $k$-exhaustive.*

*Proof.* Item (1) follows from Lemma 7: for any execution of $S$, there is an equivalent $k$-exchange, which is a $k$-match-bounded execution. Item (2) follows from Item (1) and Item (2) of Theorem 8. □