Complete Multiparty Session Type Projection with Automata

Elaine Li*[1], Felix Stutz*†[2], Thomas Wies[1], and Damien Zufferey[3]

[1] New York University `efl9013@nyu.edu, wies@cs.nyu.edu`
[2] Max Planck Institute for Software Systems `fstutz@mpi-sws.org`
[3] SonarSource `damien.zufferey@sonarsource.com`

Abstract. Multiparty session types (MSTs) are a type-based approach to verifying communication protocols. Central to MSTs is a *projection operator*: a partial function that maps protocols represented as global types to correct-by-construction implementations for each participant, represented as a communicating state machine. Existing projection operators are syntactic in nature, and trade efficiency for completeness. We present the first projection operator that is sound, complete, and efficient. Our projection separates synthesis from checking implementability. For synthesis, we use a simple automata-theoretic construction; for checking implementability, we present succinct conditions that summarize insights into the property of implementability. We use these conditions to show that MST implementability is PSPACE-complete. This improves upon a previous decision procedure that is in EXPSPACE and applies to a smaller class of MSTs. We demonstrate the effectiveness of our approach using a prototype implementation, which handles global types not supported by previous work without sacrificing performance.

Keywords: Protocol verification · Multiparty session types · Communicating state machines · Protocol fidelity · Deadlock freedom.

1 Introduction

Communication protocols are key components in many safety and operation critical systems, making them prime targets for formal verification. Unfortunately, most verification problems for such protocols (e.g. deadlock freedom) are undecidable [11]. To make verification computationally tractable, several restrictions have been proposed [2, 3, 10, 14, 33, 42]. In particular, multiparty session types (MSTs) [24] have garnered a lot of attention in recent years (see, e.g., the survey by Ancona et al. [6]). In the MST setting, a protocol is specified as a global type, which describes the desired interactions of all roles involved in the protocol. Local implementations describe behaviors for each individual role. The implementability problem for a global type asks whether there exists a collection of

*equal contribution
†corresponding author

local implementations whose composite behavior when viewed as a communicating state machine (CSM) matches that of the global type and is deadlock-free. The synthesis problem is to compute such an implementation from an implementable global type.

MST-based approaches typically solve synthesis and implementability simultaneously via an efficient syntactic *projection operator* [18,24,34,41]. Abstractly, a projection operator is a partial map from global types to collections of implementations. A projection operator `proj` is sound when every global type $\mathbf{G}$ in its domain is implemented by $\texttt{proj}(\mathbf{G})$, and complete when every implementable global type is in its domain. Existing practical projection operators for MSTs are all incomplete (or unsound). Recently, the implementability problem was shown to be decidable for a class of MSTs via a reduction to safe realizability of globally cooperative high-level message sequence charts (HMSCs) [38]. In principle, this result yields a complete and sound projection operator for the considered class. However, this operator would not be practical. In particular, the proposed implementability check is in EXPSPACE.

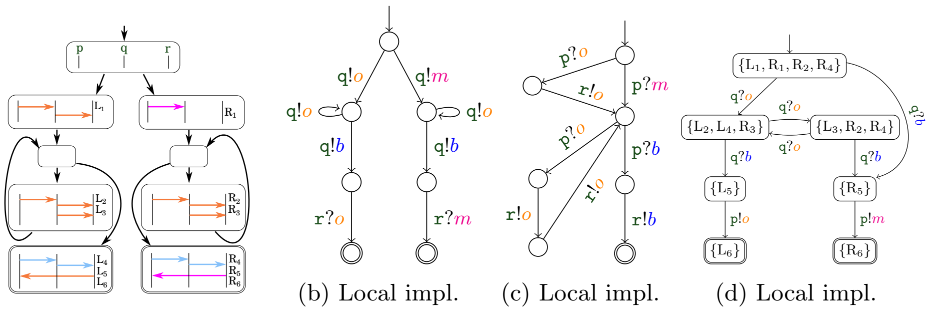
Contributions. In this paper, we present the first practical sound and complete projection operator for general MSTs. The synthesis problem for implementable global types is conceptually easy [38] – the challenge lies in determining whether a global type *is* implementable. We thus separate synthesis from checking implementability. We first use a standard automata-theoretic construction to obtain a candidate implementation for a potentially non-implementable global type. However, unlike [38], we then verify the correctness of this implementation directly using efficiently checkable conditions derived from the global type. When a global type is not implementable, our constructive completeness proof provides a counterexample trace.

The resulting projection operator yields a PSPACE decision procedure for implementability. In fact, we show that the implementability problem is PSPACE-complete. These results both generalize and tighten the decidability and complexity results obtained in [38].

We evaluate a prototype of our projection algorithm on benchmarks taken from the literature. Our prototype benefits from both the efficiency of existing lightweight but incomplete syntactic projection operators [18,24,34,41], and the generality of heavyweight automata-based model checking techniques [28,36]: it handles protocols rejected by previous practical approaches while preserving the efficiency that makes MST-based techniques so attractive.

2 Motivation and Overview

Incompleteness of existing projection operators. A key limitation of existing projection operators is that the implementation for each role is obtained via a linear traversal of the global type, and thus shares its structure. The following example, which is not projectable by any existing approach, demonstrates how enforcing structural similarity can lead to incompleteness.



(a) Odd-even protocol (b) Local impl. for role $\mathtt{p}$ (c) Local impl. for role $\mathtt{q}$ (d) Local impl. for role $\mathtt{r}$

Fig. 1: Odd-even: An implementable but not (yet) projectable protocol and its local implementations

Example 2.1 (Odd-even). Consider the following global type $\mathbf{G}_{oe}$:

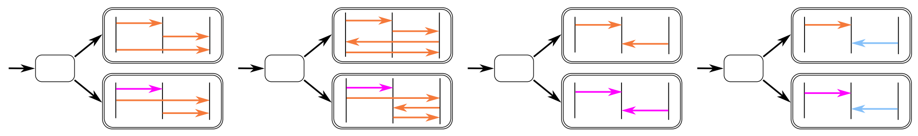
$$+\begin{cases}\mathtt{p}\to\mathtt{q}:o.\,\mathtt{q}\to\mathtt{r}:o.\,\mu t_1.\,(\mathtt{p}\to\mathtt{q}:o.\,\mathtt{q}\to\mathtt{r}:o.\,\mathtt{q}\to\mathtt{r}:o.\,t_1\ +\ \mathtt{p}\to\mathtt{q}:b.\,\mathtt{q}\to\mathtt{r}:b.\,\mathtt{r}\to\mathtt{p}:o.\,0)\\ \mathtt{p}\to\mathtt{q}:m.\,\mu t_2.\,(\mathtt{p}\to\mathtt{q}:o.\,\mathtt{q}\to\mathtt{r}:o.\,\mathtt{q}\to\mathtt{r}:o.\,t_2\ +\ \mathtt{p}\to\mathtt{q}:b.\,\mathtt{q}\to\mathtt{r}:b.\,\mathtt{r}\to\mathtt{p}:m.\,0)\end{cases}$$

A term $\mathtt{p}\to\mathtt{q}:m$ specifies the exchange of message m between sender $\mathtt{p}$ and receiver $\mathtt{q}$. The term represents two local events observed separately due to asynchrony: a send event $\mathtt{p}\triangleright\mathtt{q}!m$ observed by role $\mathtt{p}$, and a receive event $\mathtt{q}\triangleleft\mathtt{p}?m$ observed by role $\mathtt{q}$. The $+$ operator denotes choice, $\mu t.\,G$ denotes recursion, and 0 denotes protocol termination.

Fig. 1a visualizes $\mathbf{G}_{oe}$ as an HMSC. The left and right sub-protocols respectively correspond to the top and bottom branches of the protocol. Role $\mathtt{p}$ chooses a branch by sending either o or m to $\mathtt{q}$. On the left, $\mathtt{q}$ echoes this message to $\mathtt{r}$. Both branches continue in the same way: $\mathtt{p}$ sends an arbitrary number of o messages to $\mathtt{q}$, each of which is forwarded twice from $\mathtt{q}$ to $\mathtt{r}$. Role $\mathtt{p}$ signals the end of the loop by sending b to $\mathtt{q}$, which $\mathtt{q}$ forwards to $\mathtt{r}$. Finally, depending on the branch, $\mathtt{r}$ must send o or m to $\mathtt{p}$.

Figs. 1b and 1c depict the structural similarity between the global type $\mathbf{G}_{oe}$ and the implementations for $\mathtt{p}$ and $\mathtt{q}$. For the "choicemaker" role $\mathtt{p}$, the reason is evident. Role $\mathtt{q}$'s implementation collapses the continuations of both branches in the protocol into a single sub-component. For $\mathtt{r}$ (Fig. 1d), the situation is more complicated. Role $\mathtt{r}$ does not decide on or learn directly which branch is taken, but can deduce it from the parity of the number of o messages received from $\mathtt{q}$: odd means left and even means right. The resulting local implementation features transitions going back and forth between the two branches that do not exist in the global type. Syntactic projection operators fail to create such transitions. ◀

One response to the brittleness of existing projection operators has been to give up on global type specifications altogether and instead revert to model checking user-provided implementations [28, 36]. We posit that what needs rethinking is not the concept of global types, but rather how projections are computed and how implementability is checked.



(a) $\mathbf{G}_r$ (b) $\mathbf{G}'_r$ (c) $\mathbf{G}_s$ (d) $\mathbf{G}'_s$

Fig. 2: High-level message sequence charts for the global types of Example 2.2.

Our automata-theoretic approach. The synthesis step in our projection operator uses textbook automata-theoretic constructions. From a given global type, we derive a finite state machine, and use it to define a homomorphism automaton for each role. We then determinize this homomorphism automaton via subset construction to obtain a local candidate implementation for each role. If the global type is implementable, this construction always yields an implementation. The implementations shown in Figs. 1b to 1d are the result of applying this construction to $\mathbf{G}_{oe}$ from Example 2.1. Notice that the state labels in Fig. 1d correspond to sets of labels in the global protocol.

Unfortunately, not all global types are implementable.

Example 2.2. Consider the following four global types also depicted in Fig. 2:

$$\mathbf{G}_r = + \begin{cases} \mathtt{p}\to\mathtt{q}:o.\ \mathtt{q}\to\mathtt{r}:o.\ \mathtt{p}\to\mathtt{r}:o.\ 0 \\ \mathtt{p}\to\mathtt{q}:m.\ \mathtt{p}\to\mathtt{r}:o.\ \mathtt{q}\to\mathtt{r}:o.\ 0 \end{cases} \qquad \mathbf{G}_s = + \begin{cases} \mathtt{p}\to\mathtt{q}:o.\ \mathtt{r}\to\mathtt{q}:o.\ 0 \\ \mathtt{p}\to\mathtt{q}:m.\ \mathtt{r}\to\mathtt{q}:m.\ 0 \end{cases}$$

$$\mathbf{G}'_r = + \begin{cases} \mathtt{p}\to\mathtt{q}:o.\ \mathtt{q}\to\mathtt{r}:o.\ \mathtt{r}\to\mathtt{p}:o.\ \mathtt{p}\to\mathtt{r}:o.\ 0 \\ \mathtt{p}\to\mathtt{q}:m.\ \mathtt{p}\to\mathtt{r}:o.\ \mathtt{r}\to\mathtt{q}:o.\ \mathtt{q}\to\mathtt{r}:o.\ 0 \end{cases} \qquad \mathbf{G}'_s = + \begin{cases} \mathtt{p}\to\mathtt{q}:o.\ \mathtt{r}\to\mathtt{q}:b.\ 0 \\ \mathtt{p}\to\mathtt{q}:m.\ \mathtt{r}\to\mathtt{q}:b.\ 0 \end{cases}$$

Similar to $\mathbf{G}_{oe}$, in all four examples, $\mathtt{p}$ chooses a branch by sending either o or m to $\mathtt{q}$. The global type $\mathbf{G}_r$ is not implementable because $\mathtt{r}$ cannot learn which branch was chosen by $\mathtt{p}$. For any local implementation of $\mathtt{r}$ to be able to execute both branches, it must be able to receive o from $\mathtt{p}$ and $\mathtt{q}$ in any order. Because the two send events $\mathtt{p}\triangleright\mathtt{r}!o$ and $\mathtt{q}\triangleright\mathtt{r}!o$ are independent of each other, they may be reordered. Consequently, any implementation of $\mathbf{G}_r$ would have to permit executions that are consistent with global behaviors not described by $\mathbf{G}_r$, such as $\mathtt{p}\to\mathtt{q}:m.\ \mathtt{q}\to\mathtt{r}:o.\ \mathtt{p}\to\mathtt{r}:o$. Contrast this with $\mathbf{G}'_r$, which is implementable. In the top branch of $\mathbf{G}'_r$, role $\mathtt{p}$ can only send to $\mathtt{r}$ after it has received from $\mathtt{r}$, which prevents the reordering of the send events $\mathtt{p}\triangleright\mathtt{r}!o$ and $\mathtt{q}\triangleright\mathtt{r}!o$. The bottom branch is symmetric. Hence, $\mathtt{r}$ learns $\mathtt{p}$'s choice based on which message it receives first.

For the global type $\mathbf{G}_s$, role $\mathtt{r}$ again cannot learn the branch chosen by $\mathtt{p}$. That is, $\mathtt{r}$ cannot know whether to send o or m to $\mathtt{q}$, leading inevitably to deadlocking executions. In contrast, $\mathbf{G}'_s$ is again implementable because the expected behavior of $\mathtt{r}$ is independent of the choice by $\mathtt{p}$. ◀

These examples show that the implementability question is non-trivial. To check implementability, we present conditions that precisely characterize when the subset construction for $\mathbf{G}$ yields an implementation.

Overview. The rest of the paper is organized as follows. §3 contains relevant definitions for our work. §4 describes the synthesis step of our projection. §5 presents the two conditions that characterize implementability of a given global type. In §6, we prove soundness of our projection via a stronger inductive invariant guaranteeing per-role agreement on a global run of the protocol. In §7, we prove completeness by showing that our two conditions hold if a global type is implementable. In §8, we discuss the complexity of our construction and condition checks. §9 presents our artifact and evaluation, and §10 as well as §11 discuss related work.

3 Preliminaries

Words. Let Σ be a finite alphabet. Σ^* denotes the set of finite words over Σ, Σ^ω the set of infinite words, and Σ^∞ their union $\Sigma^* \cup \Sigma^\omega$. A word $u \in \Sigma^*$ is a *prefix* of word $v \in \Sigma^\infty$, denoted $u \leq v$, if there exists $w \in \Sigma^\infty$ with $u \cdot w = v$.

Message Alphabet. Let $\mathcal{P}$ be a set of roles and $\mathcal{V}$ be a set of messages. We define the set of *synchronous events* $\Sigma_{sync} := \{\mathtt{p} \to \mathtt{q} : m \mid \mathtt{p}, \mathtt{q} \in \mathcal{P} \text{ and } m \in \mathcal{V}\}$ where $\mathtt{p} \to \mathtt{q} : m$ denotes that message m is sent by $\mathtt{p}$ to $\mathtt{q}$ atomically. This is split for *asynchronous events.* For a role $\mathtt{p} \in \mathcal{P}$, we define the alphabet $\Sigma_{\mathtt{p},!} = \{\mathtt{p} \triangleright \mathtt{q}!m \mid \mathtt{q} \in \mathcal{P},\ m \in \mathcal{V}\}$ of *send* events and the alphabet $\Sigma_{\mathtt{p},?} = \{\mathtt{p} \triangleleft \mathtt{q}?m \mid \mathtt{q} \in \mathcal{P},\ m \in \mathcal{V}\}$ of *receive* events. The event $\mathtt{p} \triangleright \mathtt{q}!m$ denotes role $\mathtt{p}$ sending a message m to $\mathtt{q}$, and $\mathtt{p} \triangleleft \mathtt{q}?m$ denotes role $\mathtt{p}$ receiving a message m from $\mathtt{q}$. We write $\Sigma_{\mathtt{p}} = \Sigma_{\mathtt{p},!} \cup \Sigma_{\mathtt{p},?}$, $\Sigma_! = \bigcup_{\mathtt{p} \in \mathcal{P}} \Sigma_{\mathtt{p},!}$, and $\Sigma_? = \bigcup_{\mathtt{p} \in \mathcal{P}} \Sigma_{\mathtt{p},?}$. Finally, $\Sigma_{async} = \Sigma_! \cup \Sigma_?$. We say that $\mathtt{p}$ is *active* in $x \in \Sigma_{async}$ if $x \in \Sigma_{\mathtt{p}}$. For each role $\mathtt{p} \in \mathcal{P}$, we define a homomorphism $\Downarrow_{\Sigma_{\mathtt{p}}}$, where $x\Downarrow_{\Sigma_{\mathtt{p}}} = x$ if $x \in \Sigma_{\mathtt{p}}$ and ε otherwise. We write $\mathcal{V}(w)$ to project the send and receive events in w onto their messages. We fix $\mathcal{P}$ and $\mathcal{V}$ in the rest of the paper.

Global Types – Syntax. Global types for MSTs [31] are defined by the grammar:

$$G ::= 0 \ \mid\ \sum_{i \in I} \mathtt{p} \to \mathtt{q}_i : m_i . G_i \ \mid\ \mu t.\, G \ \mid\ t$$

where $\mathtt{p}, \mathtt{q}_i$ range over $\mathcal{P}$, m_i over $\mathcal{V}$, and t over a set of recursion variables.

We require each branch of a choice to be distinct: $\forall i, j \in I.\, i \neq j \Rightarrow (\mathtt{q}_i, m_i) \neq (\mathtt{q}_j, m_j)$, the sender and receiver of an atomic action to be distinct: $\forall i \in I.\, \mathtt{p} \neq \mathtt{q}_i$, and recursion to be guarded: in $\mu t.\, G$, there is at least one message between μt and each t in G. When $|I| = 1$, we omit $\sum$. For readability, we sometimes use the infix operator $+$ for choice, instead of $\sum$. When working with a protocol described by a global type, we write $\mathbf{G}$ to refer to the top-level type, and we use G to refer to its subterms. For the size of a global type, we disregard multiple occurrences of the same subterm.

We use the extended definition of global types from [31] that allows a sender to send messages to different roles in a choice. We call this *sender-driven choice*, as in [38], while it was called generalized choice in [31]. This definition subsumes classical MSTs that only allow *directed choice* [24]. The types we use focus on communication primitives and omit features like delegation or parametrization. We defer a detailed discussion of different MST frameworks to §11.

Global Types – Semantics. As a basis for the semantics of a global type $\mathbf{G}$, we construct a finite state machine $\mathsf{GAut}(\mathbf{G}) = (Q_{\mathbf{G}}, \Sigma_{sync}, \delta_{\mathbf{G}}, q_{0,\mathbf{G}}, F_{\mathbf{G}})$ where

- $Q_{\mathbf{G}}$ is the set of all syntactic subterms in $\mathbf{G}$ together with the term 0,
- $\delta_{\mathbf{G}}$ is the smallest set containing $(\sum_{i \in I} \mathtt{p} \to \mathtt{q}_i : m_i.G_i, \mathtt{p} \to \mathtt{q}_i : m_i, G_i)$ for each $i \in I$, as well as $(\mu t.G', \varepsilon, G')$ and $(t, \varepsilon, \mu t.G')$ for each subterm $\mu t.G'$,
- $q_{0,\mathbf{G}} = \mathbf{G}$ and $F_{\mathbf{G}} = \{0\}$.

We define a homomorphism `split` onto the asynchronous alphabet:

$$\mathtt{split}(\mathtt{p} \to \mathtt{q} : m) := \mathtt{p} \triangleright \mathtt{q}!m.\, \mathtt{q} \triangleleft \mathtt{p}?m \ .$$

The semantics $\mathcal{L}(\mathbf{G})$ of a global type $\mathbf{G}$ is given by $\mathcal{C}^{\sim}(\mathtt{split}(\mathcal{L}(\mathsf{GAut}(\mathbf{G}))))$ where $\mathcal{C}^{\sim}$ is the closure under the indistinguishability relation $\sim$ [31]. Two events are independent if they are not related by the *happened-before* relation [26]. For instance, any two send events from distinct senders are independent. Two words are indistinguishable if one can be reordered into the other by repeatedly swapping consecutive independent events. The full definition is in Appendix A.2.

Communicating State Machine [11]. $\mathcal{A} = \{\!\{A_{\mathtt{p}}\}\!\}_{\mathtt{p} \in \mathcal{P}}$ is a CSM over $\mathcal{P}$ and $\mathcal{V}$ if $A_{\mathtt{p}}$ is a finite state machine over $\Sigma_{\mathtt{p}}$ for every $\mathtt{p} \in \mathcal{P}$, denoted by $(Q_{\mathtt{p}}, \Sigma_{\mathtt{p}}, \delta_{\mathtt{p}}, q_{0,\mathtt{p}}, F_{\mathtt{p}})$. Let $\prod_{\mathtt{p} \in \mathcal{P}} s_{\mathtt{p}}$ denote the set of global states and $\mathsf{Chan} = \{(\mathtt{p}, \mathtt{q}) \mid \mathtt{p}, \mathtt{q} \in \mathcal{P}, \mathtt{p} \neq \mathtt{q}\}$ denote the set of channels. A *configuration* of $\mathcal{A}$ is a pair $(\vec{s}, \xi)$, where $\vec{s}$ is a global state and $\xi : \mathsf{Chan} \to \mathcal{V}^*$ is a mapping from each channel to a sequence of messages. We use $\vec{s}_{\mathtt{p}}$ to denote the state of $\mathtt{p}$ in $\vec{s}$. The CSM transition relation, denoted $\to$, is defined as follows.

- $(\vec{s}, \xi) \xrightarrow{\mathtt{p} \triangleright \mathtt{q}!m} (\vec{s}\,', \xi')$ if $(\vec{s}_{\mathtt{p}}, \mathtt{p} \triangleright \mathtt{q}!m, \vec{s}\,'_{\mathtt{p}}) \in \delta_{\mathtt{p}}$, $\vec{s}_{\mathtt{r}} = \vec{s}\,'_{\mathtt{r}}$ for every role $\mathtt{r} \neq \mathtt{p}$, $\xi'(\mathtt{p}, \mathtt{q}) = \xi(\mathtt{p}, \mathtt{q}) \cdot m$ and $\xi'(c) = \xi(c)$ for every other channel $c \in \mathsf{Chan}$.
- $(\vec{s}, \xi) \xrightarrow{\mathtt{q} \triangleleft \mathtt{p}?m} (\vec{s}\,', \xi')$ if $(\vec{s}_{\mathtt{q}}, \mathtt{q} \triangleleft \mathtt{p}?m, \vec{s}\,'_{\mathtt{q}}) \in \delta_{\mathtt{q}}$, $\vec{s}_{\mathtt{r}} = \vec{s}\,'_{\mathtt{r}}$ for every role $\mathtt{r} \neq \mathtt{q}$, $\xi(\mathtt{p}, \mathtt{q}) = m \cdot \xi'(\mathtt{p}, \mathtt{q})$ and $\xi'(c) = \xi(c)$ for every other channel $c \in \mathsf{Chan}$.

In the initial configuration $(\vec{s}_0, \xi_0)$, each role's state in $\vec{s}_0$ is the initial state $q_{0,\mathtt{p}}$ of $A_{\mathtt{p}}$, and ξ_0 maps each channel to ε. A configuration $(\vec{s}, \xi)$ is said to be *final* iff $\vec{s}_{\mathtt{p}}$ is final for every $\mathtt{p}$ and ξ maps each channel to ε. Runs and traces are defined in the expected way. A run is *maximal* if either it is finite and ends in a final configuration, or it is infinite. The language $\mathcal{L}(\mathcal{A})$ of the CSM $\mathcal{A}$ is defined as the set of maximal traces. A configuration $(\vec{s}, \xi)$ is a *deadlock* if it is not final and has no outgoing transitions. A CSM is *deadlock-free* if no reachable configuration is a deadlock.

Finally, implementability is formalized as follows.

Definition 3.1 (Implementability [31]). *A global type* $\mathbf{G}$ *is* implementable *if there exists a CSM* $\{\!\{A_{\mathtt{p}}\}\!\}_{\mathtt{p} \in \mathcal{P}}$ *such that the following two properties hold: (i)* protocol fidelity*:* $\mathcal{L}(\{\!\{A_{\mathtt{p}}\}\!\}_{\mathtt{p} \in \mathcal{P}}) = \mathcal{L}(\mathbf{G})$*, and (ii)* deadlock freedom*:* $\{\!\{A_{\mathtt{p}}\}\!\}_{\mathtt{p} \in \mathcal{P}}$ *is deadlock-free. We say that* $\{\!\{A_{\mathtt{p}}\}\!\}_{\mathtt{p} \in \mathcal{P}}$ *implements* $\mathbf{G}$*.*

4 Synthesizing Implementations

The construction is carried out in two steps. First, for each role $\mathtt{p} \in \mathcal{P}$, we define an intermediate state machine $\mathsf{GAut}(\mathbf{G})\!\downarrow_{\mathtt{p}}$ that is a homomorphism of $\mathsf{GAut}(\mathbf{G})$. We call $\mathsf{GAut}(\mathbf{G})\!\downarrow_{\mathtt{p}}$ the *projection by erasure* for $\mathtt{p}$, defined below.

Definition 4.1 (Projection by Erasure). *Let* $\mathbf{G}$ *be some global type with its state machine* $\mathsf{GAut}(\mathbf{G}) = (Q_{\mathbf{G}}, \Sigma_{sync}, \delta_{\mathbf{G}}, q_{0,\mathbf{G}}, F_{\mathbf{G}})$*. For each role* $\mathtt{p} \in \mathcal{P}$*, we define the state machine* $\mathsf{GAut}(\mathbf{G})\!\downarrow_{\mathtt{p}} = \ (Q_{\mathbf{G}}, \Sigma_{\mathtt{p}} \uplus \{\varepsilon\}, \delta_{\downarrow}, q_{0,\mathbf{G}}, F_{\mathbf{G}})$ *where* $\delta_{\downarrow} := \{q \xrightarrow{\mathit{split}(a)\Downarrow_{\Sigma_{\mathtt{p}}}} q' \mid q \xrightarrow{a} q' \in \delta_{\mathbf{G}}\}$*. By definition of* $\mathit{split}(\text{-})$*, it holds that* $\mathit{split}(a)\Downarrow_{\Sigma_{\mathtt{p}}} \in \Sigma_{\mathtt{p}} \uplus \{\varepsilon\}$*.*

Then, we determinize $\mathsf{GAut}(\mathbf{G})\!\downarrow_{\mathtt{p}}$ via a standard subset construction to obtain a deterministic local state machine for $\mathtt{p}$.

Definition 4.2 (Subset Construction). *Let* $\mathbf{G}$ *be a global type and* $\mathtt{p}$ *be a role. Then, the* subset construction *for* $\mathtt{p}$ *is defined as*

$$\mathscr{C}(\mathbf{G}, \mathtt{p}) = \big(Q_{\mathtt{p}}, \Sigma_{\mathtt{p}}, \delta_{\mathtt{p}}, s_{0,\mathtt{p}}, F_{\mathtt{p}}\big) \text{ where}$$

- $\delta(s, a) := \{q' \in Q_{\mathbf{G}} \mid \exists q \in s, q \xrightarrow{a}\xrightarrow{\varepsilon}{}^{*} q' \in \delta_{\downarrow}\}$*, for every* $s \subseteq Q_{\mathbf{G}}$ *and* $a \in \Sigma_{\mathtt{p}}$
- $s_{0,\mathtt{p}} := \{q \in Q_{\mathbf{G}} \mid q_{0,\mathbf{G}} \xrightarrow{\varepsilon}{}^{*} q \in \delta_{\downarrow}\}$,
- $Q_{\mathtt{p}} := \mathrm{lfp}^{\subseteq}_{\{s_{0,\mathtt{p}}\}} \lambda Q.\, Q \cup \{\delta(s, a) \mid s \in Q \wedge a \in \Sigma_{\mathtt{p}}\} \setminus \{\emptyset\}$ *, and*
- $\delta_{\mathtt{p}} := \delta|_{Q_{\mathtt{p}} \times \Sigma_{\mathtt{p}}}$
- $F_{\mathtt{p}} := \{s \in Q_{\mathtt{p}} \mid s \cap F_{\mathbf{G}} \neq \emptyset\}$

Note that the construction ensures that $Q_{\mathtt{p}}$ only contains subsets of $Q_{\mathbf{G}}$ whose states are reachable via the same traces, i.e. we typically have $|Q_{\mathtt{p}}| \ll 2^{|Q_{\mathbf{G}}|}$.

The following characterization is immediate from the subset construction; the proof can be found in Appendix B.1.

Lemma 4.3. *Let* $\mathbf{G}$ *be a global type,* $\mathtt{r}$ *be a role, and* $\mathscr{C}(\mathbf{G}, \mathtt{r})$ *be its* subset construction*. If* w *is a trace of* $\mathsf{GAut}(\mathbf{G})$*,* $\mathit{split}(w)\Downarrow_{\Sigma_{\mathtt{r}}}$ *is a trace of* $\mathscr{C}(\mathbf{G}, \mathtt{r})$*. If* u *is a trace of* $\mathscr{C}(\mathbf{G}, \mathtt{r})$*, there is a trace* w *of* $\mathsf{GAut}(\mathbf{G})$ *such that* $\mathit{split}(w)\Downarrow_{\Sigma_{\mathtt{r}}} = u$*. It holds that* $\mathcal{L}(\mathbf{G})\Downarrow_{\Sigma_{\mathtt{r}}} = \mathcal{L}(\mathscr{C}(\mathbf{G}, \mathtt{r}))$*.*

Using this lemma, we show that the CSM $\{\!\{\mathscr{C}(\mathbf{G}, \mathtt{p})\}\!\}_{\mathtt{p} \in \mathcal{P}}$ preserves all behaviors of $\mathbf{G}$.

Lemma 4.4. *For all global types* $\mathbf{G}$*,* $\mathcal{L}(\mathbf{G}) \subseteq \mathcal{L}(\{\!\{\mathscr{C}(\mathbf{G}, \mathtt{p})\}\!\}_{\mathtt{p} \in \mathcal{P}})$*.*

We briefly sketch the proof here. Given that $\{\!\{\mathscr{C}(\mathbf{G}, \mathtt{p})\}\!\}_{\mathtt{p} \in \mathcal{P}}$ is deterministic, to prove language inclusion it suffices to prove the inclusion of the respective prefix sets:

$$\mathrm{pref}(\mathcal{L}(\mathbf{G})) \subseteq \mathrm{pref}(\mathcal{L}\{\!\{\mathscr{C}(\mathbf{G}, \mathtt{p})\}\!\}_{\mathtt{p} \in \mathcal{P}})$$

Let w be a word in $\mathcal{L}(\mathbf{G})$. If w is finite, membership in $\mathcal{L}(\{\!\{\mathscr{C}(\mathbf{G}, \mathtt{p})\}\!\}_{\mathtt{p} \in \mathcal{P}})$ is immediate from the claim above. If w is infinite, we show that w has an infinite run in $\{\!\{\mathscr{C}(\mathbf{G}, \mathtt{p})\}\!\}_{\mathtt{p}\in\mathcal{P}}$ using König's Lemma. We construct an infinite graph $\mathcal{G}_w(V, E)$ with $V := \{v_\rho \mid \mathtt{trace}(\rho) \leq w\}$ and $E := \{(v_{\rho_1}, v_{\rho_2}) \mid \exists\, x \in \Sigma_{async}.\ \mathtt{trace}(\rho_2) = \mathtt{trace}(\rho_1) \cdot x\}$. Because $\{\!\{\mathscr{C}(\mathbf{G}, \mathtt{p})\}\!\}_{\mathtt{p}\in\mathcal{P}}$ is deterministic, $\mathcal{G}_w$ is a tree rooted at v_ε, the vertex corresponding to the empty run. By König's Lemma, every infinite tree contains either a vertex of infinite degree or an infinite path. Because $\{\!\{\mathscr{C}(\mathbf{G}, \mathtt{p})\}\!\}_{\mathtt{p}\in\mathcal{P}}$ consists of a finite number of communicating state machines, the last configuration of any run has a finite number of next configurations, and $\mathcal{G}_w$ is finitely branching. Therefore, there must exist an infinite path in $\mathcal{G}_w$ representing an infinite run for w, and thus $w \in \mathcal{L}(\{\!\{\mathscr{C}(\mathbf{G}, \mathtt{p})\}\!\}_{\mathtt{p}\in\mathcal{P}})$.

The proof of the inclusion of prefix sets proceeds by structural induction and primarily relies on Lemma 4.3 and the fact that all prefixes in $\mathcal{L}(\mathbf{G})$ respect the order of send before receive events.

5 Checking Implementability

We now turn our attention to checking implementability of a CSM produced by the subset construction. We revisit the global types from Example 2.2 (also shown in Fig. 2), which demonstrate that the naive subset construction does not always yield a sound implementation. From these examples, we distill our conditions that precisely identify the implementable global types.

In general, a global type $\mathbf{G}$ is not implementable when the agreement on a global run of $\mathsf{GAut}(\mathbf{G})$ among all participating roles cannot be conveyed via sending and receiving messages alone. When this happens, roles can take locally permitted transitions that commit to incompatible global runs, resulting in a trace that is not specified by $\mathbf{G}$. Consequently, our conditions need to ensure that when a role $\mathtt{p}$ takes a transition in $\mathscr{C}(\mathbf{G}, \mathtt{p})$, it only commits to global runs that are consistent with the local views of all other roles. We discuss the relevant conditions imposed on send and receive transitions separately.

Send Validity. Consider $\mathbf{G}_s$ from Example 2.2. The CSM $\{\!\{\mathscr{C}(\mathbf{G}_s, \mathtt{p})\}\!\}_{\mathtt{p}\in\mathcal{P}}$ has an execution with the trace $\mathtt{p}\triangleright\mathtt{q}!o\cdot\mathtt{q}\triangleleft\mathtt{p}?o\cdot\mathtt{r}\triangleright\mathtt{q}!m$. This trace is possible because the initial state of $\mathscr{C}(\mathbf{G}_s, \mathtt{r})$, $s_{0,\mathtt{r}}$, contains two states of $\mathsf{GAut}(\mathbf{G}_s){\Downarrow}_{\mathtt{r}}$, each of which has a single outgoing send transition labeled with $\mathtt{r}\triangleright\mathtt{q}!o$ and $\mathtt{r}\triangleright\mathtt{q}!m$ respectively. Both of these transitions are always enabled in $s_{0,\mathtt{r}}$, meaning that $\mathtt{r}$ can send $\mathtt{r} \triangleright \mathtt{q}!m$ even when $\mathtt{p}$ has chosen the top branch and $\mathtt{q}$ expects to receive o instead of m from $\mathtt{r}$. This results in a deadlock. In contrast, while the state $s_{0,\mathtt{r}}$ in $\mathscr{C}(\mathbf{G}'_s, \mathtt{r})$ likewise contains two states of $\mathsf{GAut}(\mathbf{G}'_s){\Downarrow}_{\mathtt{r}}$, each with a single outgoing send transition, now both transitions are labeled with $\mathtt{r} \triangleright \mathtt{q}!b$. These two transitions collapse to a single one in $\mathscr{C}(\mathbf{G}'_s, \mathtt{r})$. This transition is consistent with both possible local views that $\mathtt{p}$ and $\mathtt{q}$ might hold on the global run.

Intuitively, to prevent the emergence of inconsistent local views from send transitions of $\mathscr{C}(\mathbf{G}, \mathtt{p})$, we must enforce that for every state $s \in Q_{\mathtt{p}}$ with an outgoing send transition labeled x, a transition labeled x must be enabled in all states of $\mathsf{GAut}(\mathbf{G}){\Downarrow}_{\mathtt{p}}$ represented by s. We use the following auxiliary definition to formalize this intuition subsequently.

Definition 5.1 (Transition Origin and Destination). *Let* $s \xrightarrow{x} s' \in \delta_{\mathtt{p}}$ *be a transition in* $\mathscr{C}(\mathbf{G}, \mathtt{p})$ *and* $\delta_{\downarrow}$ *be the transition relation of* $\mathsf{GAut}(\mathbf{G}){\downarrow_{\mathtt{p}}}$. *We define the set of* transition origins $\text{tr-orig}(s \xrightarrow{x} s')$ *and* transition destinations $\text{tr-dest}(s \xrightarrow{x} s')$ *as follows:*

$$\text{tr-orig}(s \xrightarrow{x} s') := \{G \in s \mid \exists G' \in s'.\, G \xrightarrow{x}{}^{*} G' \in \delta_{\downarrow}\} \textit{ and}$$
$$\text{tr-dest}(s \xrightarrow{x} s') := \{G' \in s' \mid \exists G \in s.\, G \xrightarrow{x}{}^{*} G' \in \delta_{\downarrow}\} \ .$$

Our condition on send transitions is then stated below.

Definition 5.2 (Send Validity). $\mathscr{C}(\mathbf{G}, \mathtt{p})$ *satisfies* Send Validity *iff every send transition* $s \xrightarrow{x} s' \in \delta_{\mathtt{p}}$ *is enabled in all states contained in* s*:*

$$\forall s \xrightarrow{x} s' \in \delta_{\mathtt{p}}.\ x \in \Sigma_{\mathtt{p},!} \implies \text{tr-orig}(s \xrightarrow{x} s') = s \ .$$

Receive Validity. To motivate our condition on receive transitions, let us revisit $\mathbf{G}_r$ from Example 2.2. The CSM $\{\!\{\mathscr{C}(\mathbf{G}_r, \mathtt{p})\}\!\}_{\mathtt{p} \in \mathcal{P}}$ recognizes the following trace not in the global type language $\mathcal{L}(\mathbf{G}_r)$:

$$\mathtt{p} \triangleright \mathtt{q}!o \cdot \mathtt{q} \triangleleft \mathtt{p}?o \cdot \mathtt{q} \triangleright \mathtt{r}!o \cdot \mathtt{p} \triangleright \mathtt{r}!o \cdot \mathtt{r} \triangleleft \mathtt{p}?o \cdot \mathtt{r} \triangleleft \mathtt{q}?o \ .$$

The issue lies with $\mathtt{r}$ which cannot distinguish between the two branches in $\mathbf{G}_r$. The initial state $s_{0,\mathtt{r}}$ of $\mathscr{C}(\mathbf{G}_r, \mathtt{r})$ has two states of $\mathsf{GAut}(\mathbf{G}_r)$ corresponding to the subterms $G_t := \mathtt{q} \to \mathtt{r} : o.\, \mathtt{p} \to \mathtt{r} : o.\, 0$ and $G_b := \mathtt{p} \to \mathtt{r} : o.\, \mathtt{q} \to \mathtt{r} : o.\, 0$. Here, G_t and G_b are the top and bottom branch of $\mathbf{G}_r$ respectively. This means that there are outgoing transitions in $s_{0,\mathtt{r}}$ labeled with $\mathtt{r} \triangleleft \mathtt{p}?o$ and $\mathtt{r} \triangleleft \mathtt{q}?o$. If $\mathtt{r}$ takes the transition labeled $\mathtt{r} \triangleleft \mathtt{p}?o$, it commits to the bottom branch G_b. However, observe that the message o from $\mathtt{p}$ can also be available at this time point if the other roles follow the top branch G_t. This is because $\mathtt{p}$ can send o to $\mathtt{r}$ without waiting for $\mathtt{r}$ to first receive from $\mathtt{q}$. In this scenario, the roles disagree on which global run of $\mathsf{GAut}(\mathbf{G}_r)$ to follow, resulting in the violating trace above.

Contrast this with $\mathbf{G}'_r$. Here, $s_{0,\mathtt{r}}$ again has outgoing transitions labeled with $\mathtt{r} \triangleleft \mathtt{p}?o$ and $\mathtt{r} \triangleleft \mathtt{q}?o$. However, if $\mathtt{r}$ takes the transition labeled $\mathtt{r} \triangleleft \mathtt{p}?o$, committing to the bottom branch, no disagreement occurs. This is because if the other roles are following the top branch, then $\mathtt{p}$ is blocked from sending to $\mathtt{r}$ until after it has received confirmation that $\mathtt{r}$ has received its first message from $\mathtt{q}$.

For a receive transition $s \xrightarrow{x} s_1$ in $\mathscr{C}(\mathbf{G}, \mathtt{p})$ to be safe, we must enforce that the receive event x cannot also be available due to reordered sent messages in the continuation $G_2 \in s_2$ of another outgoing receive transition $s \xrightarrow{y} s_2$. To formalize this condition, we use the set $M^{\mathcal{B}}_{(G\ldots)}$ of *available messages* for a syntactic subterm G of $\mathbf{G}$ and a set of *blocked* roles $\mathcal{B}$. This notion was already defined in [31, Sec. 2.2]. Intuitively, $M^{\mathcal{B}}_{(G\ldots)}$ consists of all send events $\mathtt{q} \triangleright \mathtt{r}!m$ that can occur on the traces of G such that m will be the first message added to channel $(\mathtt{q}, \mathtt{r})$ before any of the roles in $\mathcal{B}$ takes a step.

Available messages. The set of available messages is recursively defined on the structure of the global type. To obtain all possible messages, we need to unfold the distinct recursion variables once. For this, we define a map $get\mu$ from variable to subterms and write $get\mu_{\mathbf{G}}$ for $get\mu(\mathbf{G})$:

$$get\mu(0) := [\,] \qquad get\mu(t) := [\,] \qquad get\mu(\mu t.G) := [t \mapsto G] \cup get\mu(G)$$
$$get\mu(\textstyle\sum_{i\in I} \mathtt{p}\to\mathtt{q}_i : m_i.G_i) := \bigcup_{i\in I} get\mu(G_i)$$

The function $M^{\mathcal{B},T}_{(\text{-}\ldots)}$ keeps a set of unfolded variables T, which is empty initially.

$$M^{\mathcal{B},T}_{(0\ldots)} := \emptyset \qquad M^{\mathcal{B},T}_{(\mu t.G\ldots)} := M^{\mathcal{B},T\cup\{t\}}_{(G\ldots)} \qquad M^{\mathcal{B},T}_{(t\ldots)} := \begin{cases} \emptyset & \text{if } t \in T \\ M^{\mathcal{B},T\cup\{t\}}_{(get\mu_{\mathbf{G}}(t)\ldots)} & \text{if } t \notin T \end{cases}$$

$$M^{\mathcal{B},T}_{(\sum_{i\in I} \mathtt{p}\to\mathtt{q}_i:m_i.G_i\ldots)} := \begin{cases} \bigcup_{i\in I, m\in\mathcal{V}} (M^{\mathcal{B},T}_{(G_i\ldots)} \setminus \{\mathtt{q}_i \triangleleft \mathtt{p}?m\}) \cup \{\mathtt{q}_i \triangleleft \mathtt{p}?m_i\} & \text{if } \mathtt{p} \notin \mathcal{B} \\ \bigcup_{i\in I} M^{\mathcal{B}\cup\{\mathtt{q}_i\},T}_{(G_i\ldots)} & \text{if } \mathtt{p} \in \mathcal{B} \end{cases}$$

We write $M^{\mathcal{B}}_{(G\ldots)}$ for $M^{\mathcal{B},\emptyset}_{(G\ldots)}$. If $\mathcal{B}$ is a singleton set, we omit set notation and write $M^{\mathtt{p}}_{(G\ldots)}$ for $M^{\{\mathtt{p}\}}_{(G\ldots)}$. The set of available messages captures the possible states of all channels before a given receive transition is taken.

Definition 5.3 (Receive Validity). *$\mathscr{C}(\mathbf{G}, \mathtt{p})$ satisfies* Receive Validity *iff no receive transition is enabled in an alternative continuation that originates from the same source state:*

$$\forall s \xrightarrow{\mathtt{p}\triangleleft\mathtt{q}_1?m_1} s_1,\, s \xrightarrow{\mathtt{p}\triangleleft\mathtt{q}_2?m_2} s_2 \in \delta_{\mathtt{p}}.$$
$$\mathtt{q}_1 \neq \mathtt{q}_2 \implies \forall\, G_2 \in \text{tr-dest}(s \xrightarrow{\mathtt{p}\triangleleft\mathtt{q}_2?m_2} s_2).\ \mathtt{q}_1 \triangleright \mathtt{p}!m_1 \notin M^{\mathtt{p}}_{(G_2\ldots)}\ .$$

Subset Projection. We are now ready to define our projection operator.

Definition 5.4 (Subset Projection of G). *The* subset projection $\mathscr{P}(\mathbf{G}, \mathtt{p})$ *of* $\mathbf{G}$ *onto* $\mathtt{p}$ *is* $\mathscr{C}(\mathbf{G}, \mathtt{p})$ *if it satisfies Send Validity and Receive Validity. We lift this operation to a partial function from global types to CSMs in the expected way.*

We conclude our discussion with an observation about the syntactic structure of the subset projection: Send Validity implies that no state has both outgoing send and receive transitions (also known as mixed choice).

Corollary 5.5 (No Mixed Choice). *If $\mathscr{P}(\mathbf{G}, \mathtt{p})$ satisfies Send Validity, then for all $s \xrightarrow{x_1} s_1, s \xrightarrow{x_2} s_2 \in \delta_{\mathtt{p}}$, $x_1 \in \Sigma_!$ iff $x_2 \in \Sigma_!$.*

6 Soundness

In this section, we prove the soundness of our subset projection, stated as follows.

Theorem 6.1. *Let* $\mathbf{G}$ *be a global type and* $\{\!\{\mathscr{P}(\mathbf{G}, \mathtt{p})\}\!\}_{\mathtt{p}\in\mathcal{P}}$ *be the subset projection. Then,* $\{\!\{\mathscr{P}(\mathbf{G}, \mathtt{p})\}\!\}_{\mathtt{p}\in\mathcal{P}}$ *implements* $\mathbf{G}$.

Recall that implementability is defined as protocol fidelity and deadlock freedom. Protocol fidelity consists of two language inclusions. The first inclusion, $\mathcal{L}(\mathbf{G}) \subseteq \mathcal{L}(\{\!\{\mathscr{P}(\mathbf{G}, \mathtt{p})\}\!\}_{\mathtt{p}\in\mathcal{P}})$, enforces that the subset projection generates at least all behaviors of the global type. We showed in Lemma 4.4 that this holds for the subset construction alone (without Send and Receive Validity).

The second inclusion, $\mathcal{L}(\{\!\{\mathscr{P}(\mathbf{G}, \mathtt{p})\}\!\}_{\mathtt{p}\in\mathcal{P}}) \subseteq \mathcal{L}(\mathbf{G})$, enforces that no new behaviors are introduced. The proof of this direction relies on a stronger inductive invariant that we show for all traces of the subset projection. As discussed in §5, violations of implementability occur when roles commit to global runs that are inconsistent with the local views of other roles. Our inductive invariant states the exact opposite: that all local views are consistent with one another. First, we formalize the local view of a role.

Definition 6.2 (Possible run sets). *Let* $\mathbf{G}$ *be a global type and* $\mathsf{GAut}(\mathbf{G})$ *be the corresponding state machine. Let* $\mathtt{p}$ *be a role and* $w \in \Sigma_{async}^*$ *be a word. We define the set of possible runs* $\mathrm{R}_{\mathtt{p}}^{\mathbf{G}}(w)$ *as all maximal runs of* $\mathsf{GAut}(\mathbf{G})$ *that are consistent with* $\mathtt{p}$*'s local view of* w*:*

$$\mathrm{R}_{\mathtt{p}}^{\mathbf{G}}(w) := \{\rho \textit{ is a maximal run of } \mathsf{GAut}(\mathbf{G}) \mid w{\Downarrow_{\Sigma_\mathtt{p}}} \leq \boldsymbol{split}(\boldsymbol{trace}(\rho)){\Downarrow_{\Sigma_\mathtt{p}}}\} \ .$$

While Definition 6.2 captures the set of maximal runs that are consistent with the local view of a single role, we would like to refer to the set of runs that is consistent with the local view of all roles. We formalize this as the intersection of the possible run sets for all roles, which we denote as

$$I(w) := \bigcap_{\mathtt{p}\in\mathcal{P}} \mathrm{R}_{\mathtt{p}}^{\mathbf{G}}(w) \ .$$

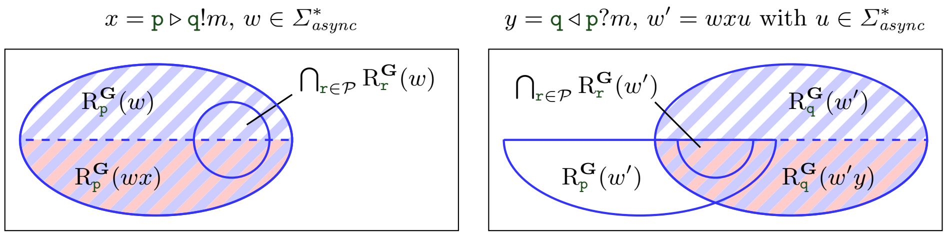
With these definitions in hand, we can now formulate our inductive invariant:

Lemma 6.3. *Let* $\mathbf{G}$ *be a global type and* $\{\!\{\mathscr{P}(\mathbf{G}, \mathtt{p})\}\!\}_{\mathtt{p}\in\mathcal{P}}$ *be the subset projection. Let* w *be a trace of* $\{\!\{\mathscr{P}(\mathbf{G}, \mathtt{p})\}\!\}_{\mathtt{p}\in\mathcal{P}}$*. It holds that* $I(w)$ *is non-empty.*

The reasoning for the sufficiency of Lemma 6.3 is included in the proof of Theorem 6.1, found in Appendix C. In the rest of this section, we focus our efforts on how to show this inductive invariant, namely that the intersection of all roles' possible run sets is non-empty.

We begin with the observation that the empty trace ε is consistent with all runs. As a result, $I(\varepsilon) = \bigcap_{\mathtt{p}\in\mathcal{P}} \mathrm{R}_{\mathtt{p}}^{\mathbf{G}}(\varepsilon)$ contains all maximal runs in $\mathsf{GAut}(\mathbf{G})$. By definition, state machines for global types include at least one run, and the base case is trivially discharged. Intuitively, $I(w)$ shrinks as more events are appended to w, but we show that at no point does it shrink to $\emptyset$. We consider the cases where a send or receive event is appended to the trace separately, and show that the intersection set shrinks in a principled way that preserves non-emptiness. In fact, when a trace is extended with a receive event, Receive Validity guarantees that the intersection set does not shrink at all.

Lemma 6.4. *Let* $\mathbf{G}$ *be a global type and* $\{\!\{\mathscr{P}(\mathbf{G}, \mathtt{p})\}\!\}_{\mathtt{p}\in\mathcal{P}}$ *be the subset projection. Let* wx *be a trace of* $\{\!\{\mathscr{P}(\mathbf{G}, \mathtt{p})\}\!\}_{\mathtt{p}\in\mathcal{P}}$ *such that* $x \in \Sigma_?$*. Then,* $I(w) = I(wx)$*.*



$x = \mathtt{p} \triangleright \mathtt{q}!m$, $w \in \Sigma^*_{async}$
$\mathrm{R}^{\mathbf{G}}_{\mathtt{p}}(w)$
$\bigcap_{\mathtt{r}\in\mathcal{P}} \mathrm{R}^{\mathbf{G}}_{\mathtt{r}}(w)$
$\mathrm{R}^{\mathbf{G}}_{\mathtt{p}}(wx)$
$y = \mathtt{q} \triangleleft \mathtt{p}?m$, $w' = wxu$ with $u \in \Sigma^*_{async}$
$\bigcap_{\mathtt{r}\in\mathcal{P}} \mathrm{R}^{\mathbf{G}}_{\mathtt{r}}(w')$
$\mathrm{R}^{\mathbf{G}}_{\mathtt{q}}(w')$
$\mathrm{R}^{\mathbf{G}}_{\mathtt{p}}(w')$
$\mathrm{R}^{\mathbf{G}}_{\mathtt{q}}(w'y)$

Fig. 3: Evolution of $\mathrm{R}^{\mathbf{G}}_{-}(\text{-})$ sets when $\mathtt{p}$ sends a message m and $\mathtt{q}$ receives it.

To prove this equality, we further refine our characterization of intersection sets. In particular, we show that in the receive case, the intersection between the sender and receiver's possible run sets stays the same, i.e.

$$\mathrm{R}^{\mathbf{G}}_{\mathtt{p}}(w) \cap \mathrm{R}^{\mathbf{G}}_{\mathtt{q}}(w) = \mathrm{R}^{\mathbf{G}}_{\mathtt{p}}(wx) \cap \mathrm{R}^{\mathbf{G}}_{\mathtt{q}}(wx) \ .$$

Note that it is not the case that the receiver only follows a subset of the sender's possible runs. In other words, $\mathrm{R}^{\mathbf{G}}_{\mathtt{q}}(w) \subseteq \mathrm{R}^{\mathbf{G}}_{\mathtt{p}}(w)$ is not inductive. The equality above simply states that a receive action can only eliminate runs that have already been eliminated by its sender. Fig. 3 depicts this relation.

Given that the intersection set strictly shrinks, the burden of eliminating runs must then fall upon send events. We show that send transitions shrink the possible run set of the sender in a way that is *prefix-preserving*. To make this more precise, we introduce the following definition on runs.

Definition 6.5 (Unique splitting of a possible run). *Let* $\mathbf{G}$ *be a global type,* $\mathtt{p}$ *a role, and* $w \in \Sigma^*_{async}$ *a word. Let* ρ *be a possible run in* $\mathrm{R}^{\mathbf{G}}_{\mathtt{p}}(w)$*. We define the longest prefix of* ρ *matching* w*:*

$$\alpha' := \max\{\rho' \mid \rho' \leq \rho \ \wedge \ \mathtt{split}(\mathtt{trace}(\rho'))\Downarrow_{\Sigma_{\mathtt{p}}} \leq w\Downarrow_{\Sigma_{\mathtt{p}}}\} \ .$$

If $\alpha' \neq \rho$*, we can split* ρ *into* $\rho = \alpha \cdot G \xrightarrow{l} G' \cdot \beta$ *where* $\alpha' = \alpha \cdot G$*,* G' *denotes the state following* G*, and* β *denotes the suffix of* ρ *following* $\alpha \cdot G \cdot G'$*. We call* $\alpha \cdot G \xrightarrow{l} G' \cdot \beta$ *the unique splitting of* ρ *for* $\mathtt{p}$ *matching* w*. We omit the role* $\mathtt{p}$ *when obvious from context. This splitting is always unique because the maximal prefix of any* $\rho \in \mathrm{R}^{\mathbf{G}}_{\mathtt{p}}(w)$ *matching* w *is unique.*

When role $\mathtt{p}$ fires a send transition $\mathtt{p} \triangleright \mathtt{q}!m$, any run $\rho = \alpha \cdot G \xrightarrow{l} G' \cdot \beta$ in $\mathtt{p}$'s possible run with $\mathtt{split}(l)\Downarrow_{\Sigma_{\mathtt{p}}} \neq \mathtt{p} \triangleright \mathtt{q}!m$ is eliminated. While the resulting possible run set could no longer contain runs that end with $G' \cdot \beta$, Send Validity guarantees that it must contain runs that begin with $\alpha \cdot G$. This is formalized by the following lemma.

Lemma 6.6. *Let* $\mathbf{G}$ *be a global type and* $\{\!\{\mathscr{P}(\mathbf{G}, \mathtt{p})\}\!\}_{\mathtt{p}\in\mathcal{P}}$ *be the subset projection. Let* wx *be a trace of* $\{\!\{\mathscr{P}(\mathbf{G}, \mathtt{p})\}\!\}_{\mathtt{p}\in\mathcal{P}}$ *such that* $x \in \Sigma_! \cap \Sigma_{\mathtt{p}}$ *for some* $\mathtt{p} \in \mathcal{P}$*. Let*

ρ be a run in $I(w)$, and $\alpha \cdot G \xrightarrow{l} G' \cdot \beta$ be the unique splitting of ρ for $\texttt{p}$ with respect to w. Then, there exists a run ρ' in $I(wx)$ such that $\alpha \cdot G \leq \rho'$.

This concludes our discussion of the send and receive cases in the inductive step to show the non-emptiness of the intersection of all roles' possible run sets. The full proofs and additional definitions can be found in Appendix C.

7 Completeness

In this section, we prove completeness of our approach. While soundness states that if a global type's subset projection is defined, it then implements the global type, completeness considers the reverse direction.

Theorem 7.1 (Completeness). *If $\mathbf{G}$ is implementable, then $\{\!\{\mathscr{P}(\mathbf{G}, \texttt{p})\}\!\}_{\texttt{p} \in \mathcal{P}}$ is defined.*

We sketch the proof and refer to Appendix D for the full proof.

From the assumption that $\mathbf{G}$ is implementable, we know there exists a witness CSM that implements $\mathbf{G}$. While the soundness proof picks our subset projection as the existential witness for showing implementability – thereby allowing us to reason directly about a particular implementation – completeness only guarantees the existence of some witness CSM. We cannot assume without loss of generality that this witness CSM is our subset construction; however, we must use the fact that it implements $\mathbf{G}$ to show that Send and Receive Validity hold on our subset construction.

We proceed via proof by contradiction: we assume the negation of Send and Receive Validity for the subset construction, and show a contradiction to the fact that this witness CSM implements $\mathbf{G}$. In particular, we contradict protocol fidelity (Definition 3.1(i)), stating that the witness CSM generates precisely the language $\mathcal{L}(\mathbf{G})$. To do so, we exploit a simulation argument: we first show that the negation of Send and Receive Validity forces the subset construction to recognize a trace that is not a prefix of any word in $\mathcal{L}(\mathbf{G})$. Then, we show that this trace must also be recognized by the witness CSM, under the assumption that the witness CSM implements $\mathbf{G}$.

To highlight the constructive nature of our proof, we convert our proof obligation to a witness construction obligation. To contradict protocol fidelity, it suffices to construct a witness trace v_0 satisfying two properties, where $\{\!\{B_{\texttt{p}}\}\!\}_{\texttt{p} \in \mathcal{P}}$ is our witness CSM:

(a) v_0 is a trace of $\{\!\{B_{\texttt{p}}\}\!\}_{\texttt{p} \in \mathcal{P}}$, and
(b) the run intersection set of v_0 is empty: $I(v_0) = \bigcap_{\texttt{p} \in \mathcal{P}} \mathrm{R}^{\mathbf{G}}_{\texttt{p}}(v_0) = \emptyset$.

We first establish the sufficiency of conditions (a) and (b). Because $\{\!\{B_{\texttt{p}}\}\!\}_{\texttt{p} \in \mathcal{P}}$ is deadlock-free by assumption, every prefix extends to a maximal trace. Thus, to prove the inequality of the two languages $\mathcal{L}(\{\!\{B_{\texttt{p}}\}\!\}_{\texttt{p} \in \mathcal{P}})$ and $\mathcal{L}(\mathbf{G})$, it suffices to prove the inequality of their respective prefix sets. In turn, it suffices to show the existence of a prefix of a word in one language that is not a prefix of any word in the other. We choose to construct a prefix in the CSM language that is not a prefix in $\mathcal{L}(\mathbf{G})$. We again leverage the definition of intersection sets (Definition 6.2) to weaken the property of language non-membership to the property of having an empty intersection set as follows. By the semantics of $\mathcal{L}(\mathbf{G})$, for any $w \in \mathcal{L}(\mathbf{G})$, there exists $w' \in \mathtt{split}(\mathcal{L}(\mathsf{GAut}(\mathbf{G})))$ with $w \sim w'$. For any $w' \in \mathtt{split}(\mathcal{L}(\mathsf{GAut}(\mathbf{G})))$, it trivially holds that w' has a non-empty intersection set. Because intersection sets are invariant under the indistinguishability relation $\sim$, w must also have a non-empty intersection set. Since intersection sets are monotonically decreasing, if the intersection set of w is non-empty, then for any $v \leq w$, the intersection set of v is also non-empty. Modus tollens of the chain of reasoning above tells us that in order to show a word is not a prefix in $\mathcal{L}(\mathbf{G})$, it suffices to show that its intersection set is empty.

Having established the sufficiency of properties (a) and (b) for our witness construction, we present the steps to construct v_0 from the negation of Send and Receive Validity respectively. We start by constructing a trace in $\{\!\{\mathscr{C}(\mathbf{G}, \mathtt{p})_\mathtt{p}\}\!\}_{\mathtt{p}\in\mathcal{P}}$ that satisfies (b), and then show that $\{\!\{B_\mathtt{p}\}\!\}_{\mathtt{p}\in\mathcal{P}}$ also recognizes the trace, thereby satisfying (a). In both cases, let $\mathtt{p}$ be the role and s be the state for which the respective validity condition is violated.

Send Validity (Definition 5.2). Let $s \xrightarrow{\mathtt{p}\triangleright\mathtt{q}!m} s' \in \delta_\mathtt{p}$ be a transition such that

$$\text{tr-orig}(s \xrightarrow{\mathtt{p}\triangleright\mathtt{q}!m} s') \neq s \ .$$

First, we find a trace u of $\{\!\{\mathscr{C}(\mathbf{G}, \mathtt{p})_\mathtt{p}\}\!\}_{\mathtt{p}\in\mathcal{P}}$ that satisfies: (1) role $\mathtt{p}$ is in state s in the CSM configuration reached via u, and (2) the run of $\mathsf{GAut}(\mathbf{G})$ on u visits a state in $s \setminus \text{tr-orig}(s \xrightarrow{\mathtt{p}\triangleright\mathtt{q}!m} s')$. We obtain such a witness u from the $\mathtt{split}(\mathtt{trace}(-))$ of a run prefix of $\mathsf{GAut}(\mathbf{G})$ that ends in some state in $s \setminus \text{tr-orig}(s \xrightarrow{\mathtt{p}\triangleright\mathtt{q}!m} s')$. Any prefix thus obtained satisfies (1) by definition of $\mathscr{C}(\mathbf{G}, \mathtt{p})$, and satisfies (2) by construction. Due to the fact that send transitions are always enabled in a CSM, $u \cdot \mathtt{p}\triangleright\mathtt{q}!m$ must also be a trace of $\{\!\{\mathscr{C}(\mathbf{G}, \mathtt{p})\}\!\}_{\mathtt{p}\in\mathcal{P}}$, thus satisfying property (a) by a simulation argument. We then argue that $u \cdot \mathtt{p}\triangleright\mathtt{q}!m$ satisfies property (b), stating that $I(u \cdot \mathtt{p}\triangleright\mathtt{q}!m)$ is empty: the negation of Send Validity gives that there exist no run extensions from our candidate state in $s \setminus \text{tr-orig}(s \xrightarrow{\mathtt{p}\triangleright\mathtt{q}!m} s')$ with the immediate next action $\mathtt{p} \to \mathtt{q} : m$, and therefore there exists no maximal run in $\mathsf{GAut}(\mathbf{G})$ consistent with $u \cdot \mathtt{p}\triangleright\mathtt{q}!m$.

Receive Validity (Definition 5.3). Let $s \xrightarrow{\mathtt{p}\triangleleft\mathtt{q}_1?m_1} s_1$ and $s \xrightarrow{\mathtt{p}\triangleleft\mathtt{q}_2?m_2} s_2 \in \delta_\mathtt{p}$ be two transitions, and let $G_2 \in \text{tr-dest}(s \xrightarrow{\mathtt{p}\triangleleft\mathtt{q}_2?m_2} s_2)$ such that

$$\mathtt{q}_1 \neq \mathtt{q}_2 \text{ and } \mathtt{q}_1 \triangleright \mathtt{p}!m_1 \in M^{\mathtt{p}}_{(G_2\ldots)} \ .$$

Constructing the witness v_0 pivots on finding a trace u of $\{\!\{\mathscr{C}(\mathbf{G}, \mathtt{p})\}\!\}_{\mathtt{p}\in\mathcal{P}}$ such that both $u\cdot\mathtt{p}\triangleleft\mathtt{q}_1?m_1$ and $u\cdot\mathtt{p}\triangleleft\mathtt{q}_2?m_2$ are traces of $\{\!\{\mathscr{C}(\mathbf{G}, \mathtt{p})\}\!\}_{\mathtt{p}\in\mathcal{P}}$. Equivalently, we show there exists a reachable configuration of $\{\!\{\mathscr{C}(\mathbf{G}, \mathtt{p})\}\!\}_{\mathtt{p}\in\mathcal{P}}$ in which $\mathtt{p}$ can receive either message from distinct senders $\mathtt{q}_1$ and $\mathtt{q}_2$. Formally, the local state of $\mathtt{p}$ has two outgoing states labeled with $\mathtt{p} \triangleleft \mathtt{q}_1?m_1$ and $\mathtt{p} \triangleleft \mathtt{q}_2?m_2$, and the channels $\mathtt{q}_1, \mathtt{p}$ and $\mathtt{q}_2, \mathtt{p}$ have m_1 and m_2 at their respective heads. We construct such a u by considering a run in $\mathsf{GAut}(\mathbf{G})$ that contains two transitions labeled with $\mathtt{q}_1 \to \mathtt{p} : m_1$ and $\mathtt{q}_2 \to \mathtt{p} : m_2$. Such a run must exist due to the negation of Receive Validity. We start with the split trace of this run, and argue that, from the definition of $M(\text{-})$ and the indistinguishability relation $\sim$, we can perform iterative reorderings using $\sim$ to bubble the send action $\mathtt{q}_1 \triangleright \mathtt{p}!m_1$ to the position before the receive action $\mathtt{p}\triangleleft\mathtt{q}_2?m_2$. Then, (a) for $u\cdot\mathtt{p}\triangleleft\mathtt{q}_1?m_1$ holds by a simulation argument. We then separately show that (b) holds for $\mathtt{p} \triangleleft \mathtt{q}_1?m_1$ using similar reasoning as the send case to complete the proof that $u \cdot \mathtt{p} \triangleleft \mathtt{q}_1?m_1$ suffices as a witness for v_0.

It is worth noting that the construction of the witness prefix v_0 in the proof immediately yields an algorithm for computing counterexample traces to implementability.

Remark 7.2 (Mixed Choice is Not Needed to Implement Global Types). Theorem 7.1 basically shows the necessity of Send Validity for implementability. Corollary 5.5 shows that Send Validity precludes states with both send and receive outgoing transitions. Together, this implies that an implementable global type can always be implemented without mixed choice. Note that the syntactic restrictions on global types do not inherently prevent mixed choice states from arising in a role's subset construction, as evidenced by $\mathtt{r}$ in the following type: $\mathtt{p}\to\mathtt{q}:l.\,\mathtt{q}\to\mathtt{r}:m.\,0 \;+\; \mathtt{p}\to\mathtt{q}:r.\,\mathtt{r}\to\mathtt{q}:m.\,0$. Our completeness result thus implies that this type is not implementable. Most MST frameworks [18,24,31] implicitly force *no mixed choice* through syntactic restrictions on local types. We are the first to prove that mixed choice states are indeed not necessary for completeness. This is interesting because mixed choice is known to be crucial for the expressive power of the synchronous π-calculus compared to its asynchronous variant [32].

8 Complexity

In this section, we establish PSPACE-completeness of checking implementability for global types.

Theorem 8.1. *The MST implementability problem is PSPACE-complete.*

Proof. We first establish the upper bound. The decision procedure enumerates for each role $\mathtt{p}$ the subsets of $\mathsf{GAut}(\mathbf{G})\!\downarrow_{\mathtt{p}}$. This can be done in polynomial space and exponential time. For each $\mathtt{p}$ and $s \subseteq Q_{\mathbf{G}}$, it then (i) checks membership of s in $Q_{\mathtt{p}}$ of $\mathscr{C}(\mathbf{G}, \mathtt{p})$, and (ii) if $s \in Q_{\mathtt{p}}$, checks whether all outgoing transitions of s in $\mathscr{C}(\mathbf{G}, \mathtt{p})$ satisfy Send and Receive Validity. Check (i) can be reduced to the intersection non-emptiness problem for nondeterministic finite state machines, which is in PSPACE [44]. It is easy to see that check (ii) can be done in polynomial time. In particular, the computation of available messages for Receive Validity only requires a single unfolding of every loop in $\mathbf{G}$.

Note that the synthesis problem has the same complexity. The subset construction to determinize $\mathsf{GAut}(\mathbf{G}){\downarrow_{\mathtt{p}}}$ can be done using a PSPACE transducer. While the output can be of exponential size, it is written on an extra tape that is not counted towards memory usage. However, this means we need to perform the validity checks as described above instead of using the computed deterministic state machines.

Second, we prove the lower bound. The proof is inspired by the proof for Theorem 4 [4] in which Alur et al. prove that checking safe realizability of bounded HMSCs is PSPACE-hard. We reduce the PSPACE-complete problem of checking universality of an NFA $M = (Q, \Delta, \delta, q_0, F)$ to checking implementability. Without loss of generality, we assume that every state can reach a final state. We construct a global type $\mathbf{G}$ for $\mathtt{p}, \mathtt{q}$ and $\mathtt{r}$ that is implementable iff $\mathcal{L}(M) = \Delta^*$. For this, we define subterms G_l and G_r as well as G_q for every $q \in Q$ and G_*. We use a fresh letter $\perp$ to handle final states of M. We also define $\mathtt{p}\leftrightarrow\mathtt{q}\!:\!m$ as an abbreviation for $\mathtt{p}\to\mathtt{q}\!:\!m \,.\, \mathtt{q}\to\mathtt{p}\!:\!m$.

$$\mathbf{G} := G_l + G_r$$

$$G_l := \mathtt{p}\leftrightarrow\mathtt{q}\!:\!l \,.\, \mathtt{p}\leftrightarrow\mathtt{r}\!:\!go \,.\, G_{q_0}$$

$$G_q := \begin{cases} \sum_{(a,q')\in\delta(q)} (\mathtt{r}\leftrightarrow\mathtt{q}\!:\!a \,.\, G_{q'}) & \text{if } q \notin F \\ \mathtt{r}\leftrightarrow\mathtt{q}\!:\!\perp .\, 0 \;+\; \sum_{(a,q')\in\delta(q)} (\mathtt{r}\leftrightarrow\mathtt{q}\!:\!a \,.\, G_{q'}) & \text{if } q \in F \end{cases}$$

$$G_r := \mathtt{p}\leftrightarrow\mathtt{q}\!:\!r \,.\, \mathtt{p}\leftrightarrow\mathtt{r}\!:\!go \,.\, G_*$$

$$G_* := \mathtt{r}\leftrightarrow\mathtt{q}\!:\!\perp .\, 0 + \sum_{a\in\Delta} (\mathtt{r}\leftrightarrow\mathtt{q}\!:\!a \,.\, G_*)$$

The global type $\mathbf{G}$ is constructed such that $\mathtt{p}$ first decides whether words from $\mathcal{L}(M)$ or from Δ^* are sent subsequently. This decision is known to $\mathtt{p}$ and $\mathtt{q}$ but not to $\mathtt{r}$. The protocol then continues with $\mathtt{r}$ sending letters from Δ to $\mathtt{q}$, and $\mathtt{p}$ is not involved. Intuitively, $\mathtt{q}$ is able to receive these letters if and only if $\mathcal{L}(M) = \Delta^*$. From Theorems 6.1 and 7.1, we know that $\{\!\{\mathscr{C}(\mathbf{G}, \mathtt{p})_{\mathtt{p}}\}\!\}_{\mathtt{p}\in\mathcal{P}}$ implements $\mathbf{G}$ if $\mathbf{G}$ is implementable.

We claim that $\{\!\{\mathscr{C}(\mathbf{G}, \mathtt{p})_{\mathtt{p}}\}\!\}_{\mathtt{p}\in\mathcal{P}}$ implements $\mathbf{G}$ if and only if $\mathcal{L}(M) = \Delta^*$.

First, assume that $\mathcal{L}(M) \neq \Delta^*$. Then, there exists $w \notin \mathcal{L}(M)$. We can construct the following run of $\{\!\{\mathscr{C}(\mathbf{G}, \mathtt{p})_{\mathtt{p}}\}\!\}_{\mathtt{p}\in\mathcal{P}}$ that deadlocks. Role $\mathtt{p}$ chooses the left subterm G_l and, subsequently, $\mathtt{r}$ sends w to $\mathtt{q}$. We do a case analysis on whether w contains a prefix w' such that $w' \notin \mathrm{pref}(\mathcal{L}(M))$. If so, sending the last letter of a minimal prefix leads to a deadlock in $\{\!\{\mathscr{C}(\mathbf{G}, \mathtt{p})_{\mathtt{p}}\}\!\}_{\mathtt{p}\in\mathcal{P}}$, contradicting deadlock freedom. If not, it holds that w is a prefix of a word in $\mathcal{L}(M)$. Still, role $\mathtt{r}$ can send $\perp$, which cannot be received, also contradicting deadlock freedom.

Second, assume that $\mathcal{L}(M) = \Delta^*$. With this, it is fine that $\mathtt{r}$ does not know the branch. Role $\mathtt{q}$ will be able to receive all messages since $\mathscr{C}(\mathbf{G}, \mathtt{q})$ can receive, letter by letter, $w.\perp$ for every $w \in \mathcal{L}(M)$ from $\mathtt{r}$. Thus, protocol fidelity and deadlock freedom hold, concluding the proof.

Note that PSPACE-hardness only holds if the size of $\mathbf{G}$ does not account for common subterms multiple times. Because every message is immediately acknowledged, the constructed global type specifies a universally 1-bounded [23] language, proving that PSPACE-hardness persists for such a restriction. For our construction, it does not hold that $\mathcal{V}(\mathcal{L}(G_l)\Downarrow_{\Sigma_{\mathtt{q},?}}) = \mathcal{L}(M)$. We chose so to have a more compact protocol. However, we can easily fix this by sending the decision of $\mathtt{r}$ first to $\mathtt{p}$, allowing to omit the messages $\perp$ to $\mathtt{q}$. □

This result and the fact that local languages are preserved by the subset projection (Lemma 4.3) leads to the following observation.

Corollary 8.2. *Let* $\mathbf{G}$ *be an implementable global type. Then, the subset projection* $\{\!\!\{\mathscr{P}(\mathbf{G},\mathtt{p})\}\!\!\}_{\mathtt{p}\in\mathcal{P}}$ *is a local language preserving implementation for* $\mathbf{G}$*, i.e.,* $\mathcal{L}(\mathscr{P}(\mathbf{G},\mathtt{p})) = \mathcal{L}(\mathbf{G})\Downarrow_{\Sigma_\mathtt{p}}$ *for every* $\mathtt{p}$*, and can be computed in PSPACE.*

Remark 8.3 (MST implementability with directed choice is PSPACE-hard). Theorem 8.1 is stated for global types with sender-driven choice but the provided type is in fact directed. Thus, the PSPACE lower bound also holds for implementability of types with directed choice.

9 Evaluation

We consider the following three aspects in the evaluation of our approach: (E1) difficulty of implementation (E2) completeness, and (E3) comparison to state of the art.

For this, we implemented our subset projection in a prototype tool [1, 37]. It takes a global type as input and computes the subset projection for each role. It was straightforward to implement the core functionality in approximately 700 lines of Python3 code closely following the formalization (E1).

We consider global types (and communication protocols) from seven different sources as well as all examples from this work (cf. 1st column of Table 1). Our experiments were run on a computer with an Intel Core i7-1165G7 CPU and used at most 100MB of memory. The results are summarized in Table 1. The reported size is the number of states and transitions of the respective state machine, which allows not to account for multiple occurrences of the same subterm. As expected, our tool can project every implementable protocol we have considered (E2).

Regarding the comparison against the state of the art (E3), we directly compared our subset projection to the incomplete approach by Majumdar et al. [31], and found that the run times are in the same order of magnitude in general (typically a few milliseconds). However, the projection of [31] fails to project four implementable protocols (including Example 2.1). We discuss some of the other examples in more detail in the next section. We further note that most of the run times reported by Scalas and Yoshida [36] on their model checking based tool are around 1 second and are thus two to three orders of magnitude slower.

Source	Name	Impl.	Subset Proj. (complete)		Size	$\lvert\mathcal{P}\rvert$	Size Proj's	[31] (incomplete)	
[35]	Instrument Contr. Prot. A	✓	✓	0.4 ms	22	3	61	✓	0.2 ms
	Instrument Contr. Prot. B	✓	✓	0.3 ms	17	3	47	✓	0.1 ms
	OAuth2	✓	✓	0.1 ms	10	3	23	✓	< 0.1 ms
[34]	Multi Party Game	✓	✓	0.5 ms	21	3	67	✓	0.1 ms
[24]	Streaming	✓	✓	0.2 ms	13	4	28	✓	< 0.1 ms
[13]	Non-Compatible Merge	✓	✓	0.2 ms	11	3	25	✓	0.1 ms
[45]	Spring-Hibernate	✓	✓	1.0 ms	62	6	118	✓	0.7 ms
[31]	Group Present	✓	✓	0.6 ms	51	4	85	✓	0.6 ms
	Late Learning	✓	✓	0.3 ms	17	4	34	✓	0.2 ms
	Load Balancer ($n = 10$)	✓	✓	3.9 ms	36	12	106	✓	2.4 ms
	Logging ($n = 10$)	✓	✓	71.5 ms	81	13	322	✓	10.0 ms
[38]	2 Buyer Protocol	✓	✓	0.5 ms	22	3	60	✓	0.2 ms
	2B-Prot. Omit No	✓	✓	0.4 ms	19	3	56	(×)	0.1 ms
	2B-Prot. Subscription	✓	✓	0.7 ms	46	3	95	(×)	0.3 ms
	2B-Prot. Inner Recursion	✓	✓	0.4 ms	17	3	51	✓	0.1 ms
New	Odd-even (Example 2.1)	✓	✓	0.5 ms	32	3	70	(×)	0.2 ms
	$\mathbf{G}_r$ – Receive Val. Violated (§2)	×	×	0.1 ms	12	3	-	(×)	< 0.1 ms
	$\mathbf{G}'_r$ – Receive Val. Satisfied (§2)	✓	✓	0.2 ms	16	3	35	✓	0.1 ms
	$\mathbf{G}_s$ – Send Val. Violated (§2)	×	×	< 0.1 ms	8	3	-	(×)	< 0.1 ms
	$\mathbf{G}'_s$ – Send Val. Satisfied (§2)	✓	✓	< 0.1 ms	7	3	17	✓	< 0.1 ms
	$\mathbf{G}_{\text{fold}}$ (§10)	✓	✓	0.4 ms	21	3	50	(×)	0.1 ms
	$\mathbf{G}_{\text{unf}}$ (§10)	✓	✓	0.4 ms	30	3	61	✓	0.2 ms

Table 1: Projecting Global Types. For every protocol, we report whether it is implementable ✓ or not ×, the time to compute our subset projection and the generalized projection by Majumdar et al. [31] as well as the outcome as ✓ for "implementable", × for "not implementable" and (×) for "not known". We also give the size of the protocol (number of states and transitions), the number of roles, the combined size of all subset projections (number of states and transitions).

10 Discussion

Success of Syntactic Projections Depends on Representation. Let us illustrate how unfolding recursion helps syntactic projection operators to succeed. Consider this implementable global type, which is not syntactically projectable:

$$\mathbf{G}_{\text{fold}} := + \begin{cases} \mathtt{p}\to\mathtt{q}:o.\ \mu t_1.\ (\mathtt{p}\to\mathtt{q}:o.\ \mathtt{q}\to\mathtt{r}:o.\ t_1\ +\ \mathtt{p}\to\mathtt{q}:b.\ \mathtt{q}\to\mathtt{r}:b.\ 0) \\ \mathtt{p}\to\mathtt{q}:m.\ \mathtt{q}\to\mathtt{r}:m.\ \mu t_2.\ (\mathtt{p}\to\mathtt{q}:o.\ \mathtt{q}\to\mathtt{r}:o.\ t_2\ +\ \mathtt{p}\to\mathtt{q}:b.\ \mathtt{q}\to\mathtt{r}:b.\ 0) \end{cases}.$$

Similar to projection by erasure, a syntactic projection erases events that a role is not involved in and immediately tries to *merge* different branches. The merge operator is a partial operator that checks sufficient conditions for implementability. Here, the merge operator fails for $\mathtt{r}$ because it cannot merge a recursion variable binder and a message reception. Unfolding the global type preserves the represented protocol and resolves this issue:

$$\mathbf{G}_{\text{unf}} := + \begin{cases} \mathtt{p}\to\mathtt{q}:o.\ \begin{cases} \mathtt{p}\to\mathtt{q}:b.\ \mathtt{q}\to\mathtt{r}:b.\ 0 \\ \mathtt{p}\to\mathtt{q}:o.\ \mathtt{q}\to\mathtt{r}:o.\ \mu t_1.\ (\mathtt{p}\to\mathtt{q}:o.\ \mathtt{q}\to\mathtt{r}:o.\ t_1\ +\ \mathtt{p}\to\mathtt{q}:b.\ \mathtt{q}\to\mathtt{r}:b.\ 0) \end{cases} \\ \mathtt{p}\to\mathtt{q}:m.\ \mathtt{q}\to\mathtt{r}:m.\ \mu t_2.\ (\mathtt{p}\to\mathtt{q}:o.\ \mathtt{q}\to\mathtt{r}:o.\ t_2\ +\ \mathtt{p}\to\mathtt{q}:b.\ \mathtt{q}\to\mathtt{r}:b.\ 0) \end{cases}.$$

(We refer to Fig. 4 in Appendix E.1 for visual representations of both global types.) This global type can be projected with most syntactic projection operators and shows that the representation of the global type matters for syntactic projectability. However, such unfolding tricks do not always work, e.g. for the odd-even protocol (Example 2.1). We avoid this brittleness using automata and separating the synthesis from checking implementability.

Entailed Properties from the Literature. We defined implementability for a global type as the question of whether there exists a deadlock-free CSM that generates the same language as the global type. Various other properties of implementations and protocols have been proposed in the literature. Here, we give a brief overview and defer to Appendix E.2 for a detailed analysis. *Progress* [18], a common property, requires that every sent message is eventually received and every expected message will eventually be sent. With deadlock freedom, our subset projection trivially satisfies progress for finite traces. For infinite traces, as expected, fairness assumptions are required to enforce progress. Similarly, our subset projection prevents *unspecified receptions* [14] and *orphan messages* [9, 21], respectively interpreted in our multiparty setting with sender-driven choice. We also ensure that every local transition of each role is *executable* [14], i.e. it is taken in some run of the CSM. Any implementation of a global type has the *stable property* [28], i.e., one can always reach a configuration with empty channels from every reachable configuration. While the properties above are naturally satisfied by our subset projection, the following ones can be checked directly on an implementable global type without explicitly constructing the implementation. A global type is *terminating* [36] iff it does not contain recursion and *never-terminating* [36] iff it does not contain term 0.

11 Related Work

MSTs were introduced by Honda et al. [24] with a process algebra semantics, and the connection to CSMs was established soon afterwards [20].

In this work, we present a complete projection procedure for global types with sender-driven choice. The work by Castagna et al. [13] is the only one to present a projection that aims for completeness. Their semantic conditions, however, are not effectively computable and their notion of completeness is “less demanding than the classical ones” [13]. They consider multiple implementations, generating different sets of traces, to be sound and complete with regard to a single global type [13, Sec. 5.3]. In addition, the algorithmic version of their conditions does not use global information as our message availability analysis does.

MST implementability relates to safe realizability of HMSCs, which is undecidable in general but decidable for certain classes [30]. Stutz [38] showed that implementability of global types that are always able to terminate is decidable.[1] The EXPSPACE decision procedure is obtained via a reduction to safe realizability of globally-cooperative HMSCs, by proving that the HMSC encoding [39]

[1]This syntactic restriction is referred to as 0-reachability in [38].

of any implementable global type is globally-cooperative and generalizing results for infinite executions. Thus, our PSPACE-completeness result both generalizes and tightens the earlier decidability result obtained in [38]. Stutz [38] also investigates how HMSC techniques for safe realizability can be applied to the MST setting – using the formal connection between MST implementability and safe realizability of HMSCs – and establishes an undecidability result for a variant of MST implementability with a relaxed indistinguishability relation.

Similar to the MST setting, there have been approaches in the HMSC literature that tie branching to a role making a choice. We refer the reader to the work by Majumdar et al. [31] for a survey.

Standard MST frameworks project a global type to a set of *local types* rather than a CSM. Local types are easily translated to FSMs [31, Def.11]. Our projection operator, though, can yield FSMs that cannot be expressed with the limited syntax of local types. Consider this implementable global type: $\mathtt{p} \to \mathtt{q} : o . 0 \ + \ \mathtt{p} \to \mathtt{q} : m . \mathtt{p} \to \mathtt{r} : b . 0$. The subset projection for $\mathtt{r}$ has two final states connected by a transition labeled $\mathtt{r} \triangleleft \mathtt{p} ? b$. In the syntax of local types, 0 is the only term indicating termination, which means that final states with outgoing transitions cannot be expressed. In contrast to the syntactic restrictions for global types, which are key to effective verification, we consider local types unnecessarily restrictive. Usually, local implementations are type-checked against their local types and subtyping gives some implementation freedom [12,16,17,27]. However, one can also view our subset projection as a local specification of the actual implementation. We conjecture that subtyping would then amount to a variation of alternating refinement [5].

CSMs are Turing-powerful [11] but decidable classes were obtained for different semantics: restricted communication topology [33,42], half-duplex communication (only for two roles) [14], input-bounded [10], and unreliable channels [2,3]. Global types (as well choreography automata [7]) can only express existentially 1-bounded, 1-synchronizable and half-duplex communication [39]. Key to this result is that sending and receiving a message is specified atomically in a global type — a feature Dagnino et al. [19] waived for their deconfined global types. However, Dagnino et al. [19] use deconfined types to capture the behavior of a given system rather than projecting to obtain a system that generates specified behaviors.

This work relies on reliable communication as is standard for MST frameworks. Work on fault-tolerant MST frameworks [8, 43] attempts to relax this restriction. In the setting of reliable communication, both context-free [25, 40] and parametric [15, 22] versions of session types have been proposed to capture more expressive protocols and entire protocol families respectively. Extending our approach to these generalizations is an interesting direction for future work.

Acknowledgements. This work is funded in part by the National Science Foundation under grant 1815633. Felix Stutz was supported by the Deutsche Forschungsgemeinschaft project 389792660 TRR 248—CPEC.

References

1. Prototype Implementation of Subset Projection for Multiparty Session Types, https://gitlab.mpi-sws.org/fstutz/async-mpst-gen-choice/
2. Abdulla, P.A., Aiswarya, C., Atig, M.F.: Data communicating processes with unreliable channels. In: Grohe, M., Koskinen, E., Shankar, N. (eds.) Proceedings of the 31st Annual ACM/IEEE Symposium on Logic in Computer Science, LICS '16, New York, NY, USA, July 5-8, 2016. pp. 166–175. ACM (2016). https://doi.org/10.1145/2933575.2934535
3. Abdulla, P.A., Bouajjani, A., Jonsson, B.: On-the-fly analysis of systems with unbounded, lossy FIFO channels. In: Hu, A.J., Vardi, M.Y. (eds.) Computer Aided Verification, 10th International Conference, CAV'98, Vancouver, BC, Canada, June 28 - July 2, 1998, Proceedings. Lecture Notes in Computer Science, vol. 1427, pp. 305–318. Springer (1998). https://doi.org/10.1007/BFb0028754
4. Alur, R., Etessami, K., Yannakakis, M.: Realizability and verification of MSC graphs. Theor. Comput. Sci. **331**(1), 97–114 (2005). https://doi.org/10.1016/j.tcs.2004.09.034
5. Alur, R., Henzinger, T.A., Kupferman, O., Vardi, M.Y.: Alternating refinement relations. In: Sangiorgi, D., de Simone, R. (eds.) CONCUR '98: Concurrency Theory, 9th International Conference, Nice, France, September 8-11, 1998, Proceedings. Lecture Notes in Computer Science, vol. 1466, pp. 163–178. Springer (1998). https://doi.org/10.1007/BFb0055622
6. Ancona, D., Bono, V., Bravetti, M., Campos, J., Castagna, G., Deniélou, P., Gay, S.J., Gesbert, N., Giachino, E., Hu, R., Johnsen, E.B., Martins, F., Mascardi, V., Montesi, F., Neykova, R., Ng, N., Padovani, L., Vasconcelos, V.T., Yoshida, N.: Behavioral types in programming languages. Found. Trends Program. Lang. **3**(2-3), 95–230 (2016). https://doi.org/10.1561/2500000031
7. Barbanera, F., Lanese, I., Tuosto, E.: Choreography automata. In: Bliudze, S., Bocchi, L. (eds.) Coordination Models and Languages - 22nd IFIP WG 6.1 International Conference, COORDINATION 2020, Held as Part of the 15th International Federated Conference on Distributed Computing Techniques, DisCoTec 2020, Valletta, Malta, June 15-19, 2020, Proceedings. Lecture Notes in Computer Science, vol. 12134, pp. 86–106. Springer (2020). https://doi.org/10.1007/978-3-030-50029-0_6
8. Barwell, A.D., Scalas, A., Yoshida, N., Zhou, F.: Generalised multiparty session types with crash-stop failures. In: Klin, B., Lasota, S., Muscholl, A. (eds.) 33rd International Conference on Concurrency Theory, CONCUR 2022, September 12-16, 2022, Warsaw, Poland. LIPIcs, vol. 243, pp. 35:1–35:25. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2022). https://doi.org/10.4230/LIPIcs.CONCUR.2022.35
9. Bocchi, L., Lange, J., Yoshida, N.: Meeting deadlines together. In: Aceto, L., de Frutos-Escrig, D. (eds.) 26th International Conference on Concurrency Theory, CONCUR 2015, Madrid, Spain, September 1.4, 2015. LIPIcs, vol. 42, pp. 283–296. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2015). https://doi.org/10.4230/LIPIcs.CONCUR.2015.283
10. Bollig, B., Finkel, A., Suresh, A.: Bounded reachability problems are decidable in FIFO machines. In: Konnov, I., Kovács, L. (eds.) 31st International Conference on Concurrency Theory, CONCUR 2020, September 1-4, 2020, Vienna, Austria (Virtual Conference). LIPIcs, vol. 171, pp. 49:1–49:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2020). https://doi.org/10.4230/LIPIcs.CONCUR.2020.49

11. Brand, D., Zafiropulo, P.: On communicating finite-state machines. J. ACM **30**(2), 323–342 (1983). `https://doi.org/10.1145/322374.322380`
12. Bravetti, M., Carbone, M., Zavattaro, G.: On the boundary between decidability and undecidability of asynchronous session subtyping. Theor. Comput. Sci. **722**, 19–51 (2018). `https://doi.org/10.1016/j.tcs.2018.02.010`
13. Castagna, G., Dezani-Ciancaglini, M., Padovani, L.: On global types and multi-party session. Log. Methods Comput. Sci. **8**(1) (2012). `https://doi.org/10.2168/LMCS-8(1:24)2012`
14. Cécé, G., Finkel, A.: Verification of programs with half-duplex communication. Inf. Comput. **202**(2), 166–190 (2005). `https://doi.org/10.1016/j.ic.2005.05.006`
15. Charalambides, M., Dinges, P., Agha, G.A.: Parameterized, concurrent session types for asynchronous multi-actor interactions. Sci. Comput. Program. **115-116**, 100–126 (2016). `https://doi.org/10.1016/j.scico.2015.10.006`
16. Chen, T., Dezani-Ciancaglini, M., Scalas, A., Yoshida, N.: On the preciseness of subtyping in session types. Log. Methods Comput. Sci. **13**(2) (2017). `https://doi.org/10.23638/LMCS-13(2:12)2017`
17. Chen, T., Dezani-Ciancaglini, M., Yoshida, N.: On the preciseness of subtyping in session types. In: Chitil, O., King, A., Danvy, O. (eds.) Proceedings of the 16th International Symposium on Principles and Practice of Declarative Programming, Kent, Canterbury, United Kingdom, September 8-10, 2014. pp. 135–146. ACM (2014). `https://doi.org/10.1145/2643135.2643138`
18. Coppo, M., Dezani-Ciancaglini, M., Padovani, L., Yoshida, N.: A gentle introduction to multiparty asynchronous session types. In: Bernardo, M., Johnsen, E.B. (eds.) Formal Methods for Multicore Programming - 15th International School on Formal Methods for the Design of Computer, Communication, and Software Systems, SFM 2015, Bertinoro, Italy, June 15-19, 2015, Advanced Lectures. Lecture Notes in Computer Science, vol. 9104, pp. 146–178. Springer (2015). `https://doi.org/10.1007/978-3-319-18941-3_4`
19. Dagnino, F., Giannini, P., Dezani-Ciancaglini, M.: Deconfined global types for asynchronous sessions. In: Damiani, F., Dardha, O. (eds.) Coordination Models and Languages - 23rd IFIP WG 6.1 International Conference, COORDINATION 2021, Held as Part of the 16th International Federated Conference on Distributed Computing Techniques, DisCoTec 2021, Valletta, Malta, June 14-18, 2021, Proceedings. Lecture Notes in Computer Science, vol. 12717, pp. 41–60. Springer (2021). `https://doi.org/10.1007/978-3-030-78142-2_3`
20. Deniélou, P., Yoshida, N.: Multiparty session types meet communicating automata. In: Seidl, H. (ed.) Programming Languages and Systems - 21st European Symposium on Programming, ESOP 2012, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2012, Tallinn, Estonia, March 24 - April 1, 2012. Proceedings. Lecture Notes in Computer Science, vol. 7211, pp. 194–213. Springer (2012). `https://doi.org/10.1007/978-3-642-28869-2_10`
21. Deniélou, P., Yoshida, N.: Multiparty compatibility in communicating automata: Characterisation and synthesis of global session types. In: Fomin, F.V., Freivalds, R., Kwiatkowska, M.Z., Peleg, D. (eds.) Automata, Languages, and Programming - 40th International Colloquium, ICALP 2013, Riga, Latvia, July 8-12, 2013, Proceedings, Part II. Lecture Notes in Computer Science, vol. 7966, pp. 174–186. Springer (2013). `https://doi.org/10.1007/978-3-642-39212-2_18`
22. Deniélou, P., Yoshida, N., Bejleri, A., Hu, R.: Parameterised multiparty session types. Log. Methods Comput. Sci. **8**(4) (2012). `https://doi.org/10.2168/LMCS-8(4:6)2012`

23. Genest, B., Kuske, D., Muscholl, A.: On communicating automata with bounded channels. Fundam. Inform. **80**(1-3), 147–167 (2007), `http://content.iospress.com/articles/fundamenta-informaticae/fi80-1-3-09`
24. Honda, K., Yoshida, N., Carbone, M.: Multiparty asynchronous session types. In: Necula, G.C., Wadler, P. (eds.) Proceedings of the 35th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2008, San Francisco, California, USA, January 7-12, 2008. pp. 273–284. ACM (2008). `https://doi.org/10.1145/1328438.1328472`
25. Keizer, A.C., Basold, H., Pérez, J.A.: Session coalgebras: A coalgebraic view on regular and context-free session types. ACM Trans. Program. Lang. Syst. **44**(3), 18:1–18:45 (2022). `https://doi.org/10.1145/3527633`
26. Lamport, L.: Time, clocks, and the ordering of events in a distributed system. Commun. ACM **21**(7), 558–565 (1978). `https://doi.org/10.1145/359545.359563`
27. Lange, J., Yoshida, N.: On the undecidability of asynchronous session subtyping. In: Esparza, J., Murawski, A.S. (eds.) Foundations of Software Science and Computation Structures - 20th International Conference, FOSSACS 2017, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2017, Uppsala, Sweden, April 22-29, 2017, Proceedings. Lecture Notes in Computer Science, vol. 10203, pp. 441–457 (2017). `https://doi.org/10.1007/978-3-662-54458-7_26`
28. Lange, J., Yoshida, N.: Verifying asynchronous interactions via communicating session automata. In: Dillig, I., Tasiran, S. (eds.) Computer Aided Verification - 31st International Conference, CAV 2019, New York City, NY, USA, July 15-18, 2019, Proceedings, Part I. Lecture Notes in Computer Science, vol. 11561, pp. 97–117. Springer (2019). `https://doi.org/10.1007/978-3-030-25540-4_6`
29. Li, E., Stutz, F., Wies, T., Zufferey, D.: Complete multiparty session type projection with automata. CoRR **abs/2305.17079** (2023). `https://doi.org/10.48550/arXiv.2305.17079`
30. Lohrey, M.: Realizability of high-level message sequence charts: closing the gaps. Theor. Comput. Sci. **309**(1-3), 529–554 (2003). `https://doi.org/10.1016/j.tcs.2003.08.002`
31. Majumdar, R., Mukund, M., Stutz, F., Zufferey, D.: Generalising projection in asynchronous multiparty session types. In: Haddad, S., Varacca, D. (eds.) 32nd International Conference on Concurrency Theory, CONCUR 2021, August 24-27, 2021, Virtual Conference. LIPIcs, vol. 203, pp. 35:1–35:24. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2021). `https://doi.org/10.4230/LIPIcs.CONCUR.2021.35`
32. Palamidessi, C.: Comparing the expressive power of the synchronous and asynchronous pi-calculi. Math. Struct. Comput. Sci. **13**(5), 685–719 (2003). `https://doi.org/10.1017/S0960129503004043`
33. Peng, W., Purushothaman, S.: Analysis of a class of communicating finite state machines. Acta Informatica **29**(6/7), 499–522 (1992). `https://doi.org/10.1007/BF01185558`
34. Scalas, A., Dardha, O., Hu, R., Yoshida, N.: A linear decomposition of multiparty sessions for safe distributed programming. In: Müller, P. (ed.) 31st European Conference on Object-Oriented Programming, ECOOP 2017, June 19-23, 2017, Barcelona, Spain. LIPIcs, vol. 74, pp. 24:1–24:31. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2017). `https://doi.org/10.4230/LIPIcs.ECOOP.2017.24`
35. Scalas, A., Yoshida, N.: Mpstk: The multiparty session types toolkit (2018), `https://doi.org/10.1145/3291638`

36. Scalas, A., Yoshida, N.: Less is more: multiparty session types revisited. Proc. ACM Program. Lang. **3**(POPL), 30:1–30:29 (2019). `https://doi.org/10.1145/3290343`
37. Stutz, F.: Artifact for "Complete Multiparty Session Type Projection with Automata" (Apr 2023). `https://doi.org/10.5281/zenodo.7878492`
38. Stutz, F.: Asynchronous multiparty session type implementability is decidable - lessons learned from message sequence charts. In: Ali, K., Salvaneschi, G. (eds.) 37th European Conference on Object-Oriented Programming, ECOOP 2023, July 17-21, 2023, Seattle, Washington, United States. LIPIcs, vol. 263, pp. 32:1–32:31. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2023). `https://doi.org/10.4230/LIPIcs.ECOOP.2023.32`
39. Stutz, F., Zufferey, D.: Comparing channel restrictions of communicating state machines, high-level message sequence charts, and multiparty session types. In: Ganty, P., Monica, D.D. (eds.) Proceedings of the 13th International Symposium on Games, Automata, Logics and Formal Verification, GandALF 2022, Madrid, Spain, September 21-23, 2022. EPTCS, vol. 370, pp. 194–212 (2022). `https://doi.org/10.4204/EPTCS.370.13`
40. Thiemann, P., Vasconcelos, V.T.: Context-free session types. In: Garrigue, J., Keller, G., Sumii, E. (eds.) Proceedings of the 21st ACM SIGPLAN International Conference on Functional Programming, ICFP 2016, Nara, Japan, September 18-22, 2016. pp. 462–475. ACM (2016). `https://doi.org/10.1145/2951913.2951926`
41. Toninho, B., Yoshida, N.: Certifying data in multiparty session types. J. Log. Algebraic Methods Program. **90**, 61–83 (2017). `https://doi.org/10.1016/j.jlamp.2016.11.005`
42. Torre, S.L., Madhusudan, P., Parlato, G.: Context-bounded analysis of concurrent queue systems. In: Ramakrishnan, C.R., Rehof, J. (eds.) Tools and Algorithms for the Construction and Analysis of Systems, 14th International Conference, TACAS 2008, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2008, Budapest, Hungary, March 29-April 6, 2008. Proceedings. Lecture Notes in Computer Science, vol. 4963, pp. 299–314. Springer (2008). `https://doi.org/10.1007/978-3-540-78800-3_21`
43. Viering, M., Hu, R., Eugster, P., Ziarek, L.: A multiparty session typing discipline for fault-tolerant event-driven distributed programming. Proc. ACM Program. Lang. **5**(OOPSLA), 1–30 (2021). `https://doi.org/10.1145/3485501`
44. Wehar, M.: On the complexity of intersection non-emptiness problems. Ph.D. thesis, University of Buffalo (2016)
45. Spring and Hibernate Transaction in Java. `https://www.uml-diagrams.org/examples/spring-hibernate-transaction-sequence-diagram-example.html`

A Additional Material for §3

A.1 Additional Definitions

Given a word $w = w_0 \ldots w_n$, we use $w[i]$ to denote the i-th symbol $w_i \in \Sigma$, and $w[0..i]$ to denote the subword between and including w_0 and w_i, $w_0 \ldots w_i$.

A.2 Indistinguishability Relation [31]

We define a family of *indistinguishability relations* $\sim_i \subseteq \Sigma^*_{async} \times \Sigma^*_{async}$ for $i \geq 0$ as follows. For all $w \in \Sigma^*$, we have $w \sim_0 w$. For $i = 1$, we define:

(1) If $\mathtt{p} \neq \mathtt{r}$, then $w.\mathtt{p} \triangleright \mathtt{q}!m.\mathtt{r} \triangleright \mathtt{s}!m'.u \;\sim_1\; w.\mathtt{r} \triangleright \mathtt{s}!m'.\mathtt{p} \triangleright \mathtt{q}!m.u$.
(2) If $\mathtt{q} \neq \mathtt{s}$, then $w.\mathtt{q} \triangleleft \mathtt{p}?m.\mathtt{s} \triangleleft \mathtt{r}?m'.u \;\sim_1\; w.\mathtt{s} \triangleleft \mathtt{r}?m'.\mathtt{q} \triangleleft \mathtt{p}?m.u$.
(3) If $\mathtt{p} \neq \mathtt{s} \wedge (\mathtt{p} \neq \mathtt{r} \vee \mathtt{q} \neq \mathtt{s})$, then $w.\mathtt{p} \triangleright \mathtt{q}!m.\mathtt{s} \triangleleft \mathtt{r}?m'.u \;\sim_1\; w.\mathtt{s} \triangleleft \mathtt{r}?m'.\mathtt{p} \triangleright \mathtt{q}!m.u$.
(4) If $|w{\Downarrow}_{\mathtt{p} \triangleright \mathtt{q}!\_}| > |w{\Downarrow}_{\mathtt{q} \triangleleft \mathtt{p}?\_}|$, then $w.\mathtt{p} \triangleright \mathtt{q}!m.\mathtt{q} \triangleleft \mathtt{p}?m'.u \;\sim_1\; w.\mathtt{q} \triangleleft \mathtt{p}?m'.\mathtt{p} \triangleright \mathtt{q}!m.u$.

Let w, w', w'' be sequences of events s.t. $w \sim_1 w'$ and $w' \sim_i w''$ for some i. Then, $w \sim_{i+1} w''$. We define $w \sim u$ if $w \sim_n u$ for some n.

It is easy to see that $\sim$ is an equivalence relation. Define $u \preceq_\sim v$ if there is $w \in \Sigma^*$ such that $u.w \sim v$. Observe that $u \sim v$ iff $u \preceq_\sim v$ and $v \preceq_\sim u$.

For infinite words $u, v \in \Sigma^\omega$, we define $u \preceq^\omega_\sim v$ if for each finite prefix u' of u, there is a finite prefix v' of v such that $u' \preceq_\sim v'$. Define $u \sim v$ iff $u \preceq^\omega_\sim v$ and $v \preceq^\omega_\sim u$.

We lift the equivalence relation $\sim$ on Σ^∞ to languages:

$$\mathcal{C}^\sim(L) = \left\{ w' \mid \bigvee \begin{array}{l} w' \in \Sigma^* \wedge \exists w \in \Sigma^*.\ w \in L \text{ and } w' \sim w \\ w' \in \Sigma^\omega \wedge \exists w \in \Sigma^\omega.\ w \in L \text{ and } w' \preceq^\omega_\sim w \end{array} \right\}$$

For the infinite case, we take the downward closure w.r.t. $\preceq^\omega_\sim$. Notice that the closure operator is asymmetric. Consider the protocol $(\mathtt{p} \triangleright \mathtt{q}!m.\mathtt{q} \triangleleft \mathtt{p}?m)^\omega$. Since we do not make any fairness assumption on scheduling, we need to include in the closure the execution where only the sender is scheduled, i.e., $(\mathtt{p} \triangleright \mathtt{q}!m)^\omega \preceq^\omega_\sim (\mathtt{p} \triangleright \mathtt{q}!m.\mathtt{q} \triangleleft \mathtt{p}?m)^\omega$.

A.3 State Machine

A *state machine* A is a 5-tuple $(Q, \Delta, \delta, q_0, F)$ where Q is a finite set of states, Δ is a finite alphabet, $\delta \subseteq Q \times (\Sigma \cup \{\varepsilon\}) \times Q$ is a transition relation, $q_0 \in Q$ is the initial state, and $F \subseteq Q$ is the set of final states. As is standard, we write $q \xrightarrow{x} q'$ for $(q, x, q') \in \delta$ and $q \xrightarrow{w}{}^* q'$ for its reflexive and transitive hull for $w \in \Delta^*$. We define the runs and traces in the standard way. A run is maximal if it is infinite or if it ends at a final state. The *language* $\mathcal{L}(A)$ is the set of (finite or infinite) maximal traces.

B Additional Material for §4

B.1 Proofs for §4

Lemma 4.3. *Let* $\mathbf{G}$ *be a global type,* $\mathtt{r}$ *be a role, and* $\mathscr{C}(\mathbf{G}, \mathtt{r})$ *be its* subset construction. *If* w *is a trace of* $\mathsf{GAut}(\mathbf{G})$*,* $\mathit{split}(w)\Downarrow_{\Sigma_\mathtt{r}}$ *is a trace of* $\mathscr{C}(\mathbf{G}, \mathtt{r})$*. If* u *is a trace of* $\mathscr{C}(\mathbf{G}, \mathtt{r})$*, there is a trace* w *of* $\mathsf{GAut}(\mathbf{G})$ *such that* $\mathit{split}(w)\Downarrow_{\Sigma_\mathtt{r}} = u$*. It holds that* $\mathcal{L}(\mathbf{G})\Downarrow_{\Sigma_\mathtt{r}} = \mathcal{L}(\mathscr{C}(\mathbf{G}, \mathtt{r}))$*.*

Proof. All claims are rather straightforward from the definitions and constructions and the proofs exploit the connection to the projection by erasure. We still spell them out to familiarize the reader with these.

We prove the first claim first. By construction, for every run ρ in $\mathsf{GAut}(\mathbf{G})$, there exists a run ρ' in the projection by erasure $\mathsf{GAut}(\mathbf{G})\downarrow_\mathtt{r}$. Let ρ be the run for trace w in $\mathsf{GAut}(\mathbf{G})$. Then, ρ is also a run in $\mathsf{GAut}(\mathbf{G})\downarrow_\mathtt{r}$ with trace $w\Downarrow_{\Sigma_\mathtt{p}}$. Since $\mathsf{GAut}(\mathbf{G})\downarrow_\mathtt{r}$ might be non-deterministic, we apply the subset construction from Definition 4.2. For the reachable states, this is equivalent to the definition by Sipser [**?**, Thm. 1.39].Thus, the constructed deterministic finite state machine can mimic any run (which is initial by definition) in $\mathsf{GAut}(\mathbf{G})\downarrow_\mathtt{r}$: for every run ρ' in $\mathsf{GAut}(\mathbf{G})\downarrow_\mathtt{r}$ with trace w', there is a run ρ'' in $\mathscr{C}(\mathbf{G}, \mathtt{r})$ with trace w'.

For the second claim, we consider a trace u of $\mathscr{C}(\mathbf{G}, \mathtt{r})$. Because of the subset construction, it holds that for every run ρ' in $\mathscr{C}(\mathbf{G}, \mathtt{r})$ with trace w', there is a run ρ'' in the projection by erasure $\mathsf{GAut}(\mathbf{G})\downarrow_\mathtt{r}$ with trace w'. By definition of the projection by erasure, a run ρ''' in $\mathsf{GAut}(\mathbf{G})$ exists with the same sequence of syntactic subterms as ρ'' and $\mathtt{split}(\mathtt{trace}(\rho'''))\Downarrow_{\Sigma_\mathtt{p}} = w'$.

From this, it easily follows that $\mathcal{L}(\mathscr{C}(\mathbf{G}, \mathtt{r})) = \mathcal{L}(\mathsf{GAut}(\mathbf{G})\downarrow_\mathtt{r})$ and, thus, $\mathcal{L}(\mathscr{C}(\mathbf{G}, \mathtt{r})) = \mathcal{L}(\mathbf{G})\Downarrow_{\Sigma_\mathtt{r}}$. ◻

Lemma 4.4. *For all global types* $\mathbf{G}$*,* $\mathcal{L}(\mathbf{G}) \subseteq \mathcal{L}(\{\!\{\mathscr{C}(\mathbf{G}, \mathtt{p})\}\!\}_{\mathtt{p}\in\mathcal{P}})$*.*

Proof. Given that $\{\!\{\mathscr{C}(\mathbf{G}, \mathtt{p})\}\!\}_{\mathtt{p}\in\mathcal{P}}$ is deterministic, to prove language inclusion it suffices to prove the inclusion of the respective prefix sets:

$$\mathrm{pref}(\mathcal{L}(\mathbf{G})) \subseteq \mathrm{pref}(\mathcal{L}\{\!\{\mathscr{C}(\mathbf{G}, \mathtt{p})\}\!\}_{\mathtt{p}\in\mathcal{P}})$$

We prove this via structural induction on w. The base case, $w = \varepsilon$, is trivial. For the inductive step, let $wx \in \mathrm{pref}(\mathcal{L}(\mathbf{G}))$. From the induction hypothesis, $w \in \mathrm{pref}(\mathcal{L}\{\!\{\mathscr{C}(\mathbf{G}, \mathtt{p})\}\!\}_{\mathtt{p}\in\mathcal{P}})$. It suffices to show that the transition labeled with x is enabled for the active role in x. Let $(\vec{s}, \xi)$ denote the $\{\!\{\mathscr{C}(\mathbf{G}, \mathtt{p})\}\!\}_{\mathtt{p}\in\mathcal{P}}$ configuration reached on w. In the case that $x \in \Sigma_!$, let $x = \mathtt{p} \triangleright \mathtt{q}!m$. The existence of an outgoing transition $\xrightarrow{\mathtt{p}\triangleright\mathtt{q}!m}$ from $\vec{s}_\mathtt{p}$ follows from the fact that $\mathcal{L}(\mathscr{C}(\mathbf{G}, \mathtt{p})) = \mathcal{L}(\mathbf{G})\Downarrow_{\Sigma_\mathtt{p}}$ (Lemma 4.3). The fact that $wx \in \mathrm{pref}(\mathcal{L}\{\!\{\mathscr{C}(\mathbf{G}, \mathtt{p})\}\!\}_{\mathtt{p}\in\mathcal{P}})$ follows immediately from this and the fact that send transitions in a CSM are always enabled. In the case that $x \in \Sigma_?$, let $x = \mathtt{q} \triangleleft \mathtt{p}?m$. We obtain an outgoing transition $\xrightarrow{\mathtt{q}\triangleleft\mathtt{p}?m}$ from $\vec{s}_\mathtt{p}$ analogously. We additionally need to show that $\xi(\mathtt{q}, \mathtt{p})$ contains m at the head. This follows from [31, Lemma 20] and the induction hypothesis. This concludes our proof of prefix set inclusion.

Let w be a word in $\mathcal{L}(\mathbf{G})$. Let $(\vec{s}, \xi)$ denote the $\{\!\{\mathscr{C}(\mathbf{G}, \mathtt{p})\}\!\}_{\mathtt{p}\in\mathcal{P}}$ configuration reached on w. In the case that w is finite, all states in $\vec{s}$ are final from Lemma 4.3 and all channels in ξ are empty from the fact that all send events in w contain matching receive events. In the case that w is infinite, we show that w has an infinite run in $\{\!\{\mathscr{C}(\mathbf{G}, \mathtt{p})\}\!\}_{\mathtt{p}\in\mathcal{P}}$ using König's Lemma. We construct an infinite graph $\mathcal{G}_w(V, E)$ with $V := \{v_\rho \mid \mathtt{trace}(\rho) \leq w\}$ and $E := \{(v_{\rho_1}, v_{\rho_2}) \mid \exists\, x \in \Sigma_{async}.\ \mathtt{trace}(\rho_2) = \mathtt{trace}(\rho_1) \cdot x\}$. Because $\{\!\{\mathscr{C}(\mathbf{G}, \mathtt{p})\}\!\}_{\mathtt{p}\in\mathcal{P}}$ is deterministic, $\mathcal{G}_w$ is a tree rooted at v_ε, the vertex corresponding to the empty run. By König's Lemma, every infinite tree contains either a vertex of infinite degree or a ray. Because $\{\!\{\mathscr{C}(\mathbf{G}, \mathtt{p})\}\!\}_{\mathtt{p}\in\mathcal{P}}$ consists of a finite number of communicating state machines, the last configuration of any run has a finite number of next configurations, and $\mathcal{G}_w$ is finitely branching. Therefore, there must exist a ray in $\mathcal{G}_w$ representing an infinite run for w, and thus $w \in \mathcal{L}(\{\!\{\mathscr{C}(\mathbf{G}, \mathtt{p})\}\!\}_{\mathtt{p}\in\mathcal{P}})$. ☐

C Additional Material for §6

The definition $M^{\mathcal{B},T}_{(G\ldots)}$ computes the set of available messages on the syntax of global types and follows the one by Majumdar et al. [31, Sec. 2.2]. For their proofs, they introduce another version, which computes these sets on the semantics of the global type using a concept called blocked languages. In Lemma 37, they prove the syntactic version always yields a superset of the semantic version. The proof easily generalizes to equality. Therefore, we use $M^{\mathcal{B},T}_{(\text{-}\ldots)}$ in our proofs and refer to their work for details.

Corollary C.1 (Intersection sets are invariant under $\sim$). *Let $\mathbf{G}$ be a global type. Let $w, w' \in \Sigma^*_{async}$ and $w \sim w'$. Then, $I(w) = I(w')$.*

Proof. It follows immediately from $w \sim w'$ that

$$\forall \mathtt{p} \in \mathcal{P}.\ w{\Downarrow_{\Sigma_\mathtt{p}}} = w'{\Downarrow_{\Sigma_\mathtt{p}}}$$

By the definition of I,

$$\forall \rho.\ \rho \in I(w) \Leftrightarrow \forall \mathtt{p} \in \mathcal{P}.\ w{\Downarrow_{\Sigma_\mathtt{p}}} \leq \mathtt{split}(\mathtt{trace}(\rho)){\Downarrow_{\Sigma_\mathtt{p}}}$$

Let ρ be a run in $\mathsf{GAut}(\mathbf{G})$. Then,

$$\begin{aligned} \rho \in I(w) &\Leftrightarrow \forall \mathtt{p} \in \mathcal{P}.\ w{\Downarrow_{\Sigma_\mathtt{p}}} \leq \mathtt{split}(\mathtt{trace}(\rho)){\Downarrow_{\Sigma_\mathtt{p}}} \\ &\Leftrightarrow \forall \mathtt{p} \in \mathcal{P}.\ w'{\Downarrow_{\Sigma_\mathtt{p}}} \leq \mathtt{split}(\mathtt{trace}(\rho)){\Downarrow_{\Sigma_\mathtt{p}}} \\ &\Leftrightarrow \rho \in I(w') \end{aligned}$$

☐

Proposition C.2 (Structural properties of $M^{\mathcal{B}}_{(G\ldots)}$). *Let $\mathcal{B} \subseteq \mathcal{P}$ and let G be a syntactic subterm of $\mathbf{G}$. Let $\mathtt{q} \triangleright \mathtt{p}!m \in M^{\mathcal{B}}_{(G\ldots)}$. Then, it holds that:*

(1) $M^{\mathcal{B}}_{(G\ldots)}$ *does not contain any events whose active role is blocked:* $\forall\ \mathtt{p} \in \mathcal{B}.\ M^{\mathcal{B}}_{(G\ldots)} \cap \Sigma_{\mathtt{p}} = \emptyset$

(2) There exists a run suffix β *such that:*
  - *i.* $G \cdot \beta$ *is the suffix of a maximal run in* $\mathsf{GAut}(\mathbf{G})$*,*
  - *ii.* $\xrightarrow{\mathtt{q} \to \mathtt{p}:m}$ *occurs in* β*, and*
  - *iii.* $\mathcal{B}$ *monotonically increases during the computation of* $M^{\mathcal{B}}_{(G\ldots)}$

Proof. Immediate from the definition of available messages in [31, Sec. 2.2].

Proposition C.3 (Correspondence between unique splittings and local states). *Let* $\mathbf{G}$ *be a global type, and* $\{\!\{\mathscr{C}(\mathbf{G}, \mathtt{p})\}\!\}_{\mathtt{p}\in\mathcal{P}}$ *be the subset construction for each role. Let* w *be a trace of* $\{\!\{\mathscr{C}(\mathbf{G}, \mathtt{p})\}\!\}_{\mathtt{p}\in\mathcal{P}}$*, and* $(\vec{s}, \xi)$ *be the CSM configuration reached on* w*. Let* $\mathtt{p}$ *be a role. Then, it holds that:*

$$\bigcup_{\rho \in \mathrm{R}^{\mathbf{G}}_{\mathtt{p}}(w)} \{G \mid \alpha \cdot G \xrightarrow{l} G' \cdot \beta \text{ is the unique splitting of } \rho \text{ matching } w\} \subseteq \vec{s}_{\mathtt{p}}$$

Proof. Let ρ be a run in $\mathrm{R}^{\mathbf{G}}_{\mathtt{p}}(w)$, and let $\alpha \cdot G \xrightarrow{l} G' \cdot \beta$ be its unique splitting for $\mathtt{p}$ matching w. It follows from the definition of unique splitting that $\mathtt{split}(\mathtt{trace}(\alpha \cdot G'))\Downarrow_{\Sigma_{\mathtt{p}}} = w\Downarrow_{\Sigma_{\mathtt{p}}}$. From the subset construction, there exists a run $s_1, \ldots, s_n$ in $\mathscr{C}(\mathbf{G}, \mathtt{p})$ such that $s_1 = s_{0,\mathtt{p}}$ and $s_1 \xrightarrow{w\Downarrow_{\Sigma_{\mathtt{p}}}}{}^{*}\ s_n$. By the definition of $Q_{\mathtt{p}}$, we know that s_n contains all global syntactic subterms in $\mathbf{G}$ that are reachable via $q_{0,\mathbf{G}} \xrightarrow{w\Downarrow_{\Sigma_{\mathtt{p}}}} \xrightarrow{\varepsilon}{}^{*}$ in $\mathsf{GAut}(\mathbf{G})_{\downarrow}$, of which G is one. Hence, $G \in s_n$. By assumption, $\mathscr{C}(\mathbf{G}, \mathtt{p})$ reached state $\vec{s}_{\mathtt{p}}$ on $w\Downarrow_{\Sigma_{\mathtt{p}}}$. Because subset constructions are deterministic, it follows that $s_n = \vec{s}_{\mathtt{p}}$. We conclude that $G \in \vec{s}_{\mathtt{p}}$. □

Proposition C.4 (No send transitions from final states in subset projection). *Let* $\mathbf{G}$ *be a global type, and* $\mathscr{P}(\mathbf{G}, \mathtt{p})$ *be the subset projection for* $\mathtt{p}$*. Let* $s \in F_{\mathtt{p}}$*, and* $s \xrightarrow{x} s' \in \delta_{\mathtt{p}}$*. Then,* $x \in \Sigma_{\mathtt{p},?}$*.*

Proof. Assume by contradiction that $x \in \Sigma_{\mathtt{p},!}$. We instantiate Send Validity with $s \xrightarrow{x} s'$ to obtain:

$$x \in \Sigma_{\mathtt{p},!} \implies \text{tr-orig}(s \xrightarrow{x} s') = s$$

By the definition of tr-orig(-), for every syntactic subterm $G \in \text{tr-orig}(s \xrightarrow{x} s')$:

$$\exists G' \in s'.\ G \xrightarrow{x}{}^{*} G' \in \delta{\restriction}_{\Sigma_{\mathtt{p}}}$$

Because s is a final state in $\mathscr{P}(\mathbf{G}, \mathtt{p})$, by definition it must contain a syntactic subterm that is a final state in $\mathsf{GAut}(\mathbf{G}){\restriction}_{\mathtt{p}}$. Because $\mathsf{GAut}(\mathbf{G}){\restriction}_{\mathtt{p}}$ and $\mathsf{GAut}(\mathbf{G})$ share the same set of final states 4.1, s must contain a syntactic subterm that is a final state in $\mathsf{GAut}(\mathbf{G})$. Let G_0 denote this final state. By the structure of $\mathsf{GAut}(\mathbf{G})$, there exists no outgoing transition from G_0. Therefore, G' does not exist. We reach a contradiction. □

Definition C.5 (G-complete words of $\{\!\{A_{\mathtt{p}}\}\!\}_{\mathtt{p}\in\mathcal{P}}$). *Let* $\mathbf{G}$ *be a global type,* $\{\!\{A_{\mathtt{p}}\}\!\}_{\mathtt{p}\in\mathcal{P}}$ *be a CSM, and* w *be a trace of* $\{\!\{A_{\mathtt{p}}\}\!\}_{\mathtt{p}\in\mathcal{P}}$. *We say* w *is* $\mathbf{G}$-complete *if for all roles* $\mathtt{p}$ *and for all runs* $\rho \in \bigcap_{\mathtt{p}\in\mathcal{P}} \mathrm{R}^{\mathbf{G}}_{\mathtt{p}}(w)$,

$$w{\Downarrow_{\Sigma_{\mathtt{p}}}} = \big(\mathit{split}(\mathit{trace}(\rho))\big){\Downarrow_{\Sigma_{\mathtt{p}}}}.$$

Definition C.6 (Channel-compliant [31]). *A word* $w \in \Sigma^{\infty}$ *is channel-compliant if for every prefix* $w' \leq w$ *and every* $\mathtt{p}, \mathtt{q} \in \mathcal{P}$, $\mathcal{V}(w'{\Downarrow_{\mathtt{q}\triangleleft\mathtt{p}?\_}}) \leq \mathcal{V}(w'{\Downarrow_{\mathtt{p}\triangleright\mathtt{q}!\_}})$.

Lemma 6.4. *Let* $\mathbf{G}$ *be a global type and* $\{\!\{\mathscr{P}(\mathbf{G},\mathtt{p})\}\!\}_{\mathtt{p}\in\mathcal{P}}$ *be the subset projection. Let* wx *be a trace of* $\{\!\{\mathscr{P}(\mathbf{G},\mathtt{p})\}\!\}_{\mathtt{p}\in\mathcal{P}}$ *such that* $x \in \Sigma_{?}$. *Then,* $I(w) = I(wx)$.

Proof. Let $x = \mathtt{p}\triangleleft\mathtt{q}?m$. Because wx is a trace of $\{\!\{\mathscr{P}(\mathbf{G},\mathtt{p})\}\!\}_{\mathtt{p}\in\mathcal{P}}$, there exists a run $(\vec{s}_0,\xi_0) \xrightarrow{w}{}^{*} (\vec{s},\xi) \xrightarrow{x} (\vec{s}\,',\xi')$ such that m is at the head of $\xi(\mathtt{p},\mathtt{q})$.

We assume that $I(w)$ is non-empty; if $I(w)$ is empty then $I(wx)$ is trivially empty. To show $I(w) = I(wx)$, it suffices to show the following claim.
Claim 1 : It holds that $\mathrm{R}^{\mathbf{G}}_{\mathtt{p}}(w) \cap \mathrm{R}^{\mathbf{G}}_{\mathtt{q}}(w) = \mathrm{R}^{\mathbf{G}}_{\mathtt{p}}(wx) \cap \mathrm{R}^{\mathbf{G}}_{\mathtt{q}}(wx)$.

We first show Claim 1's sufficiency for $I(w) = I(wx)$: By definition, $I(w) = \bigcap_{\mathtt{r}\in\mathcal{P}} \mathrm{R}^{\mathbf{G}}_{\mathtt{r}}(w) \subseteq \mathrm{R}^{\mathbf{G}}_{\mathtt{q}}(w) \cap \mathrm{R}^{\mathbf{G}}_{\mathtt{p}}(w)$. With *Claim 1*, it holds that $I(w) \subseteq \mathrm{R}^{\mathbf{G}}_{\mathtt{q}}(wx) \cap \mathrm{R}^{\mathbf{G}}_{\mathtt{p}}(wx)$. From this, it follows that $I(w) \subseteq \mathrm{R}^{\mathbf{G}}_{\mathtt{p}}(wx)$ (H1). Since $\mathtt{p}$ is the active role for the receive event x, i.e. $x \in \Sigma_{\mathtt{p}}$, it holds for any $\mathtt{r} \neq \mathtt{p}$, that $(wx){\Downarrow_{\Sigma_{\mathtt{r}}}} = w{\Downarrow_{\Sigma_{\mathtt{r}}}}$ and $\mathrm{R}^{\mathbf{G}}_{\mathtt{r}}(w) = \mathrm{R}^{\mathbf{G}}_{\mathtt{r}}(wx)$ (H2). Again, by definition of $\mathrm{R}^{\mathbf{G}}_{\_}$(-), it holds that $\mathrm{R}^{\mathbf{G}}_{\mathtt{p}}(wx) \subseteq \mathrm{R}^{\mathbf{G}}_{\mathtt{p}}(w)$ (H3).
We apply the observations to $I(wx)$:

$$\begin{aligned}
I(wx) &= \bigcap_{\mathtt{r}\in\mathcal{P}} \mathrm{R}^{\mathbf{G}}_{\mathtt{r}}(wx) \\
&\overset{(\mathrm{H2})}{=} \mathrm{R}^{\mathbf{G}}_{\mathtt{p}}(wx) \;\cap \bigcap_{\mathtt{r}\in\mathcal{P}\wedge\mathtt{r}\neq\mathtt{p}} \mathrm{R}^{\mathbf{G}}_{\mathtt{r}}(w) \\
&\overset{(\mathrm{H3})}{=} \mathrm{R}^{\mathbf{G}}_{\mathtt{p}}(wx) \;\cap\; \mathrm{R}^{\mathbf{G}}_{\mathtt{p}}(w) \;\cap \bigcap_{\mathtt{r}\in\mathcal{P}\wedge\mathtt{r}\neq\mathtt{p}} \mathrm{R}^{\mathbf{G}}_{\mathtt{r}}(w) \\
&= \mathrm{R}^{\mathbf{G}}_{\mathtt{p}}(wx) \;\cap\; I(w) \\
&\overset{(\mathrm{H1})}{=} I(w)\ .
\end{aligned}$$

Proof of Claim 1: We instantiate (H2) for role $\mathtt{q}$, which yields $\mathrm{R}^{\mathbf{G}}_{\mathtt{q}}(wx) = \mathrm{R}^{\mathbf{G}}_{\mathtt{q}}(w)$. The proof of Claim 1 therefore amounts to showing:

$$\mathrm{R}^{\mathbf{G}}_{\mathtt{p}}(w) \cap \mathrm{R}^{\mathbf{G}}_{\mathtt{q}}(w) = \mathrm{R}^{\mathbf{G}}_{\mathtt{p}}(wx) \cap \mathrm{R}^{\mathbf{G}}_{\mathtt{q}}(w)\ .$$

The right direction, i.e., $\mathrm{R}^{\mathbf{G}}_{\mathtt{p}}(wx) \subseteq \mathrm{R}^{\mathbf{G}}_{\mathtt{p}}(w)$, follows from (H3). For the left direction, i.e., $\mathrm{R}^{\mathbf{G}}_{\mathtt{p}}(w) \cap \mathrm{R}^{\mathbf{G}}_{\mathtt{q}}(w) \subseteq \mathrm{R}^{\mathbf{G}}_{\mathtt{p}}(wx)$, assume by contradiction that there exists a run ρ_0 such that

$$\rho_0 \in \mathrm{R}^{\mathbf{G}}_{\mathtt{p}}(w) \;\wedge\; \rho_0 \in \mathrm{R}^{\mathbf{G}}_{\mathtt{q}}(w) \;\wedge\; \rho_0 \notin \mathrm{R}^{\mathbf{G}}_{\mathtt{p}}(wx)\ .$$

Let ρ' be a run in $\mathrm{R}^{\mathbf{G}}_{\mathtt{p}}(w) \setminus \mathrm{R}^{\mathbf{G}}_{\mathtt{p}}(wx)$. Let $\alpha' \cdot G'_{pre} \xrightarrow{l'} G'_{post} \cdot \beta'$ be the unique splitting of ρ' for $\mathtt{p}$ matching w.

Let $\rho'_{\mathtt{p}}$ denote the largest consistent prefix of ρ' for $\mathtt{p}$; it is clear that $\rho'_{\mathtt{p}} = \alpha' \cdot G'_{pre}$. Formally,

$$\rho'_{\mathtt{p}} = max\{\rho \mid \rho \leq \rho' \ \wedge \ \big(\mathtt{split}(\mathtt{trace}(\rho))\big){\Downarrow}_{\Sigma_{\mathtt{p}}} \leq w{\Downarrow}_{\Sigma_{\mathtt{p}}}\} \ .$$

Let $\rho'_{\mathtt{q}}$ be defined analogously.

We claim that $\mathtt{q}$ is ahead of $\mathtt{p}$ in ρ', i.e. $\rho'_{\mathtt{p}} < \rho'_{\mathtt{q}}$. Intuitively, this claim follows from the half-duplex property of CSMs and the fact that $\mathtt{q}$ is the sender. Formally, Lemma 19 in [31] implies $\xi(\mathtt{q},\mathtt{p}) = u$ where $\mathcal{V}(w{\Downarrow}_{\mathtt{q}\triangleright\mathtt{p}!\_}) = \mathcal{V}(w{\Downarrow}_{\mathtt{p}\triangleleft\mathtt{q}?\_}).u$. Because $\xi(\mathtt{q},\mathtt{p})$ contains at least m by assumption, $|\mathcal{V}(w{\Downarrow}_{\mathtt{q}\triangleright\mathtt{p}!\_})| > |\mathcal{V}(w{\Downarrow}_{\mathtt{p}\triangleleft\mathtt{q}?\_})|$. Because $\mathcal{V}(w{\Downarrow}_{\mathtt{p}\triangleleft\mathtt{q}?\_}) < \mathcal{V}(w{\Downarrow}_{\mathtt{q}\triangleright\mathtt{p}!\_})$ and traces of CSMs are channel-compliant (Lemma 19, [31]), it holds that $\rho'_{\mathtt{q}}$ contains all $|\mathcal{V}(w{\Downarrow}_{\mathtt{p}\triangleleft\mathtt{q}?\_})|$ transition labels of the form $\mathtt{q} \to \mathtt{p} : \text{-}$ that are contained in $\rho'_{\mathtt{q}}$, plus at least one more of the form $\mathtt{q} \to \mathtt{p} : m$. Because both $\rho'_{\mathtt{p}}$ and $\rho'_{\mathtt{q}}$ are prefixes of ρ', it must be the case that $\rho'_{\mathtt{p}} < \rho'_{\mathtt{q}}$. This concludes the proof of the above claim.

By assumption, $\rho' \notin \mathrm{R}^{\mathbf{G}}_{\mathtt{p}}(wx)$ and therefore $l' \neq \mathtt{q} \to \mathtt{p} : m$. By the definition of unique splittings, $\mathtt{p}$ must be the active role in l'; by Corollary 5.5, $\mathtt{p}$ must be the receiving role in l'. In other words, l' must be of the form $\mathtt{r} \to \mathtt{p} : m'$, where either $\mathtt{r} \neq \mathtt{q}$ or $m' \neq m$.

Case: $\mathtt{r} = \mathtt{q}$ and $m' \neq m$.

We discharge this case by showing a contradiction to the assumption that m is at the head of the channel between $\mathtt{q}$ and $\mathtt{p}$.

Because $\alpha' \cdot G'_{pre} \leq \rho'_{\mathtt{p}}$ and $\rho'_{\mathtt{p}} < \rho'_{\mathtt{q}}$ from the claim above, it must be the case that $\alpha' \cdot G'_{pre} \xrightarrow{l'} G'_{post} \leq \rho'_{\mathtt{q}}$ and $\mathtt{q} \triangleright \mathtt{p}!m'$ is in $w{\Downarrow}_{\Sigma_{\mathtt{q}}}$. From Lemma 19 [31], it follows that $\mathcal{V}(w{\Downarrow}_{\mathtt{q}\triangleright\mathtt{p}!\_}) = \mathcal{V}(w{\Downarrow}_{\mathtt{p}\triangleleft\mathtt{q}?\_}).m'.u'$ and $\xi(\mathtt{q},\mathtt{p}) = m'.u'$, i.e. m' is at the head of the channel between $\mathtt{q}$ and $\mathtt{p}$. This contradicts the assumption that m is at the head of $\xi(\mathtt{p},\mathtt{q})$.

Case: $\mathtt{r} \neq \mathtt{q}$.

We discharge this case by showing a contradiction to Receive Validity. We instantiate Receive Validity with $\vec{s}_{\mathtt{p}} \xrightarrow{x} \vec{s}'_{\mathtt{p}}$ to obtain

$$\forall\, \vec{s}_{\mathtt{p}} \xrightarrow{\mathtt{p}\triangleleft\mathtt{q}_2?m_2} s_2 \in \delta_{\mathtt{p}}.\ \mathtt{q} \neq \mathtt{q}_2 \implies \forall\, G_2 \in \text{tr-dest}(\vec{s}_{\mathtt{p}} \xrightarrow{\mathtt{p}\triangleleft\mathtt{q}_2?m_2} s_2).\ \mathtt{q}\triangleright\mathtt{p}!m \notin M^{\mathtt{p}}_{(G_2\ldots)} \ .$$

We prove the negation, stated as follows:

$$\mathtt{q} \neq \mathtt{r} \wedge \exists\, s_2 \in Q_{\mathtt{p}}, G_2 \in \text{tr-dest}(\vec{s}_{\mathtt{p}} \xrightarrow{\mathtt{p}\triangleleft\mathtt{r}?m'} s_2).\ \mathtt{q} \triangleright \mathtt{p}!m \in M^{\mathtt{p}}_{(G_2\ldots)} \ .$$

The left conjunct follows immediately. From the existence of ρ' and Lemma 4.3, there exists an s_2 such that $\vec{s}_{\mathtt{p}} \xrightarrow{\mathtt{p}\triangleleft\mathtt{r}?m'} s_2 \in \delta_{\mathtt{p}}$. The fact that $G'_{post} \in \text{tr-dest}(\vec{s}_{\mathtt{p}} \xrightarrow{\mathtt{p}\triangleleft\mathtt{r}?m'} s_2)$ is trivial from the unique splitting of ρ' for $\mathtt{p}$ matching w:

$$\rho' = \alpha' \cdot G'_{pre} \xrightarrow{\mathtt{r}\to\mathtt{p}:m'} G'_{post} \cdot \beta' \ .$$

Therefore, all that remains is to show that $\mathtt{q} \triangleright \mathtt{p}!m \in M^{\mathtt{p}}_{(G'_{post} \cdots)}$. Because $\rho' \in \mathrm{R}^{\mathbf{G}}_{\mathtt{q}}(w)$ and $\alpha' \cdot G'_{pre} \xrightarrow{l'} G'_{post} \leq \rho'_{\mathtt{q}}$, where $\mathtt{q}$ is not the active role in l', there must exist a transition labeled $\mathtt{q} \to \mathtt{p} : m$ that occurs in the suffix $G'_{post} \cdot \beta'$ of ρ'. Let $G_0 \xrightarrow{\mathtt{q} \to \mathtt{p}:m} G'_0$ be the earliest occurrence of such a transition in the suffix, then:

$$\rho'_{\mathtt{q}} = \alpha' \cdot G'_{pre} \xrightarrow{l'} G'_{post} \ldots G_0 \xrightarrow{\mathtt{q} \to \mathtt{p}:m} G'_0 \cdots \ .$$

Note that G_0 must be a syntactic subterm of G'_{post}. In order for $\mathtt{q} \triangleright \mathtt{p}!m \in M^{\mathtt{p}}_{(G'_{post} \cdots)}$. to hold, it suffices to show that $\mathtt{q} \notin \mathcal{B}$ in the recursive call to $M^{\mathcal{B}}_{(G_0 \cdots)}$.

We argue this from the definition of M and the fact that $\rho'_{\mathtt{p}} = \alpha' \cdot G'_{pre}$. Suppose for the sake of contradiction that $\mathtt{q} \in \mathcal{B}$. Because M only adds receivers of already blocked senders to $\mathcal{B}$ and $M^{\mathtt{p}}_{(G'_{post} \cdots)}$ starts with $\mathcal{B} = \{\mathtt{p}\}$, there must exist a chain of message exchanges $\mathtt{s}_{i+1} \to \mathtt{s}_i : m_i$ in G'_{post} with $1 \leq i < n$, $\mathtt{p} = \mathtt{s}_n$, and $\mathtt{q} = \mathtt{s}_1$. That is, $G'_{post} \cdot \beta'$ must be of the form

$$G'_{post} \ldots G_{n-1} \xrightarrow{\mathtt{p} \to \mathtt{s}_{n-1}:m_{n-1}} G'_{n-1} \ldots G_1 \xrightarrow{\mathtt{s}_2 \to \mathtt{q}:m_1} G'_1 \ldots G_0 \xrightarrow{\mathtt{q} \to \mathtt{p}:m} G'_0 \cdots \ .$$

Let $m_0 = m$ and $\mathtt{s}_0 = \mathtt{p}$. We show by induction over i that for all $i \in [1, n]$

$$\alpha' \cdot G'_{pre} \xrightarrow{l'} G'_{post} \ldots G_i \xrightarrow{\mathtt{s}_i \to \mathtt{s}_{i-1}:m_{i-1}} G'_i \ \leq \ \rho'_{\mathtt{s}_i} \ .$$

We then obtain the desired contradiction with the fact that $\rho'_{\mathtt{s}_n} = \rho'_{\mathtt{p}} = \alpha' \cdot G'_{pre}$.

The base case of the induction follows immediately from the construction. For the induction step, assume that

$$\alpha' \cdot G'_{pre} \xrightarrow{l'} G'_{post} \ldots G_i \xrightarrow{\mathtt{s}_i \to \mathtt{s}_{i-1}:m_{i-1}} G'_i \ \leq \ \rho'_{\mathtt{s}_i} \ .$$

From the definition of $\rho'_{\mathtt{s}_i}$ and the fact that $\mathtt{s}_i$ is the active role in $\mathtt{s}_i \triangleleft \mathtt{s}_{i+1}?m_i$, it follows that $\mathtt{s}_i \triangleleft \mathtt{s}_{i+1}?m_i \in w$. Hence, we must also have $\mathtt{s}_{i+1} \triangleright \mathtt{s}_i!m_i \in w$. Since $\mathtt{s}_{i+1}$ is the active role in $\mathtt{s}_{i+1} \triangleright \mathtt{s}_i!m_i$, we can conclude

$$\alpha' \cdot G'_{pre} \xrightarrow{l'} G'_{post} \ldots G_i \xrightarrow{\mathtt{s}_{i+1} \to \mathtt{s}_i:m_i} G'_{i+1} \ \leq \ \rho'_{\mathtt{s}_{i+1}} \ .$$

□

Lemma 6.6. *Let* $\mathbf{G}$ *be a global type and* $\{\!\{\mathscr{P}(\mathbf{G}, \mathtt{p})\}\!\}_{\mathtt{p} \in \mathcal{P}}$ *be the subset projection. Let* wx *be a trace of* $\{\!\{\mathscr{P}(\mathbf{G}, \mathtt{p})\}\!\}_{\mathtt{p} \in \mathcal{P}}$ *such that* $x \in \Sigma_! \cap \Sigma_{\mathtt{p}}$ *for some* $\mathtt{p} \in \mathcal{P}$. *Let* ρ *be a run in* $I(w)$, *and* $\alpha \cdot G \xrightarrow{l} G' \cdot \beta$ *be the unique splitting of* ρ *for* $\mathtt{p}$ *with respect to* w. *Then, there exists a run* ρ' *in* $I(wx)$ *such that* $\alpha \cdot G \leq \rho'$.

Proof. Let $x = \mathtt{p} \triangleright \mathtt{q}!m$. We prove the claim by induction on the length of w.

Base Case. $w = \varepsilon$. By definition, $I(\varepsilon)$ contains all maximal runs in $\mathsf{GAut}(\mathbf{G})$, and the unique splitting prefix of any run $\rho \in I(\varepsilon)$ for $\mathtt{p}$ with respect to ε is ε. Because ε is a prefix of any run, we need only show the non-emptiness of $I(x)$. By 4.3, $\mathcal{L}(\mathbf{G}){\Downarrow}_{\Sigma_{\mathtt{p}}} = \mathcal{L}(\mathscr{P}(\mathbf{G}, \mathtt{p}))$. Because x is the prefix of a word in $\mathcal{L}(\mathscr{P}(\mathbf{G}, \mathtt{p}))$, there exists $w' \in \mathcal{L}(\mathbf{G})$ such that $x \leq w'{\Downarrow}_{\Sigma_{\mathtt{p}}}$. By the semantics of $\mathcal{L}(\mathbf{G})$, there exists a run $\rho' \in \mathsf{GAut}(\mathbf{G})$ such that x is the first symbol in $\mathtt{split}(\mathtt{trace}(\rho')){\Downarrow}_{\Sigma_{\mathtt{p}}}$, and therefore $\rho' \in I(x)$.

Induction Step. Let wx be an extension of w by $x \in \Sigma_!$.

Let ρ be a run in $I(w)$, and let $\alpha \cdot G \xrightarrow{l} G' \cdot \beta$ be the unique splitting of ρ for role $\mathtt{p}$ with respect to w. To re-establish the induction hypothesis, we need to show the existence of a run $\bar{\rho}$ in $I(wx)$ such that $\alpha \cdot G \leq \bar{\rho}$. Since $\mathtt{p}$ is the active role in x, it holds for any $\mathtt{r} \neq \mathtt{p}$ that $\mathrm{R}^{\mathbf{G}}_{\mathtt{r}}(w) = \mathrm{R}^{\mathbf{G}}_{\mathtt{r}}(wx)$. Therefore, to prove the existential claim, it suffices to construct a run $\bar{\rho}$ that satisfies:

1. $\bar{\rho} \in \mathrm{R}^{\mathbf{G}}_{\mathtt{p}}(wx)$,
2. $\bar{\rho} \in I(w)$, and
3. $\alpha \cdot G \leq \bar{\rho}$.

In the case that $l\Downarrow_{\Sigma_{\mathtt{p}}} = x$, we are done: Property 3 and 2 hold by construction, and Property 1 holds by the definition of possible run sets. In the case that $l\Downarrow_{\Sigma_{\mathtt{p}}} \neq x$, we show the existence of a transition label and state $\xrightarrow{\bar{l}} \bar{G}'$, and a maximal suffix $\bar{\beta}$ such that $\alpha \cdot G \xrightarrow{\bar{l}} \bar{G}' \cdot \bar{\beta}$ satisfies all three conditions.

Let $(\vec{s}_w, \xi_w)$ denote the CSM configuration reached on w: $(\vec{s}_0, \xi_0) \xrightarrow{w}{}^{*} (\vec{s}_w, \xi_w)$ Send Validity states that every transition in $\vec{s}_{w,\mathtt{p}}$ originates in all global states in $\vec{s}_{w,\mathtt{p}}$. By assumption, $\mathtt{p} \triangleright \mathtt{q}!m$ is a transition in $\vec{s}_{w,\mathtt{p}}$. By Proposition C.3, $\rho \in I \subseteq \mathrm{R}^{\mathbf{G}}_{\mathtt{p}}(w)$, and therefore $G \in \vec{s}_{w,\mathtt{p}}$. Therefore, Send Validity gives the existence of some $\bar{G}' \in Q_{\mathsf{GAut}(\mathbf{G})}$ such that $G \xrightarrow{\mathtt{p}\rightarrow\mathtt{q}:m} \bar{G}' \in \delta_{\mathsf{GAut}(\mathbf{G})}$. Because $\alpha \cdot G$ is a run in $\mathsf{GAut}(\mathbf{G})$ and $G \xrightarrow{\mathtt{p}\rightarrow\mathtt{q}:m} \bar{G}'$ is a transition in $\mathsf{GAut}(\mathbf{G})$, $\alpha \cdot G \xrightarrow{\mathtt{p}\rightarrow\mathtt{q}:m} \bar{G}'$ is a run in $\mathsf{GAut}(\mathbf{G})$.

The construction thus far satisfies Property 1 and 3 regardless of our choice of maximal suffix: for all choices of $\bar{\beta}$ such that $\alpha \cdot G \xrightarrow{\mathtt{p}\rightarrow\mathtt{q}:m} \bar{G}' \cdot \bar{\beta}$ is a maximal run, $wx\Downarrow_{\Sigma_{\mathtt{p}}} \leq \mathtt{split}(\mathtt{trace}(\alpha \cdot G \xrightarrow{\mathtt{p}\rightarrow\mathtt{q}:m} \bar{G}' \cdot \bar{\beta}))\Downarrow_{\Sigma_{\mathtt{p}}}$ and $\alpha \cdot G \leq \alpha \cdot G \xrightarrow{\mathtt{p}\rightarrow\mathtt{q}:m} \bar{G}' \cdot \bar{\beta}$.

Property 2, however, requires that the projection of w onto each role is consistent with $\bar{\rho}$, and this cannot be ensured by the prefix alone.

We construct the remainder of $\bar{\rho}$ by picking an arbitrary maximal suffix to form a candidate run, and iteratively performing suffix replacements on the candidate run until it lands in I. Let $\bar{\beta}$ be a run suffix such that $\alpha \cdot G \xrightarrow{\mathtt{p}\rightarrow\mathtt{q}:m} \bar{G}' \cdot \bar{\beta}$ is a maximal run in $\mathsf{GAut}(\mathbf{G})$. Let ρ_c denote our candidate run $\alpha \cdot G \xrightarrow{\mathtt{p}\rightarrow\mathtt{q}:m} \bar{G}' \cdot \bar{\beta}$. If $\rho_c \in I$, we are done. Otherwise, $\rho_c \notin I$ and there exists a non-empty set of processes $\mathcal{S} \subseteq \mathcal{P}$ such that for each $\mathtt{r} \in \mathcal{S}$,

$$w\Downarrow_{\Sigma_{\mathtt{r}}} \not\leq \mathtt{split}(\mathtt{trace}(\rho_c))\Downarrow_{\Sigma_{\mathtt{r}}} \ . \tag{1}$$

By the fact that $\rho \in I$,

$$w\Downarrow_{\Sigma_{\mathtt{r}}} \leq \mathtt{split}(\mathtt{trace}(\rho))\Downarrow_{\Sigma_{\mathtt{r}}} \ . \tag{2}$$

We can rewrite (1) and (2) above as:

$$w\Downarrow_{\Sigma_{\mathtt{r}}} \not\leq \mathtt{split}(\mathtt{trace}(\alpha \cdot G \xrightarrow{\mathtt{p}\rightarrow\mathtt{q}:m} \bar{G}' \cdot \bar{\beta}))\Downarrow_{\Sigma_{\mathtt{r}}} \tag{3}$$

$$w\Downarrow_{\Sigma_{\mathtt{r}}} \leq \mathtt{split}(\mathtt{trace}(\alpha \cdot G \xrightarrow{l} G' \cdot \beta))\Downarrow_{\Sigma_{\mathtt{r}}} \ . \tag{4}$$

By the definition of unique splitting, $\mathtt{p}$ is the active role in l. By Lemma 4.3, $\mathcal{L}(\mathbf{G})\Downarrow_{\Sigma_\mathtt{p}} = \mathcal{L}(\mathscr{P}(\mathbf{G},\mathtt{p}))$, and because $\mathtt{split}(\mathtt{trace}(\rho)) \in \mathcal{L}(\mathbf{G})$, it holds that $\mathtt{split}(\mathtt{trace}(\rho))\Downarrow_{\Sigma_\mathtt{p}} \in \mathcal{L}(\mathbf{G})\Downarrow_{\Sigma_\mathtt{p}}$, and $\mathtt{split}(\mathtt{trace}(\rho))\Downarrow_{\Sigma_\mathtt{p}} \in \mathcal{L}(\mathscr{P}(\mathbf{G},\mathtt{p}))$. By assumption, $\mathscr{P}(\mathbf{G},\mathtt{p})$ is in state $\vec{s}_{w,\mathtt{p}}$ upon consuming $w\Downarrow_{\Sigma_\mathtt{p}}$. Then, there must exist an outgoing transition from $\vec{s}_{w,\mathtt{p}}$ labeled with $\mathtt{split}(l)\Downarrow_{\Sigma_\mathtt{p}}$. By No Mixed Choice (Corollary 5.5), all outgoing transitions from $\vec{s}_{w,\mathtt{p}}$ must be send actions. Therefore, l must be of the form $\mathtt{p}\to\mathtt{q}' : m'$. By assumption, $\mathtt{q}' \neq \mathtt{q} \vee m' \neq m$.

We can further rewrite (3) and (4) to make explicit their shared prefix:

$$w\Downarrow_{\Sigma_\mathtt{r}} \not\leq (\mathtt{split}(\mathtt{trace}(\alpha \cdot G)).\ \mathtt{p} \triangleright \mathtt{q}!m.\ \mathtt{q} \triangleleft \mathtt{p}?m.\ \mathtt{split}(\mathtt{trace}(\bar{\beta})))\Downarrow_{\Sigma_\mathtt{r}} \tag{5}$$

$$w\Downarrow_{\Sigma_\mathtt{r}} \leq (\mathtt{split}(\mathtt{trace}(\alpha \cdot G)).\ \mathtt{p} \triangleright \mathtt{q}'!m'.\ \mathtt{q}' \triangleleft \mathtt{p}?m'.\ \mathtt{split}(\mathtt{trace}(\beta)))\Downarrow_{\Sigma_\mathtt{r}} \tag{6}$$

It is clear that in order for both (5) and (6) to hold, it must be the case that $\mathtt{split}(\mathtt{trace}(\alpha \cdot G))\Downarrow_{\Sigma_\mathtt{r}} \leq w\Downarrow_{\Sigma_\mathtt{r}}$.

We formalize the point of disagreement between $w\Downarrow_{\Sigma_\mathtt{r}}$ and ρ_c using an index $i_\mathtt{r}$ representing the position of the first disagreeing transition label in $\mathtt{trace}(\rho_c)$:

$$i_\mathtt{r} := \max\{i \mid \mathtt{split}(\mathtt{trace}(\rho_c[0..i-1]))\Downarrow_{\Sigma_\mathtt{r}} \leq w\Downarrow_{\Sigma_\mathtt{r}}\} \ .$$

Then, $\mathtt{split}(\mathtt{trace}(\rho_c[i_\mathtt{r}]))\Downarrow_{\Sigma_\mathtt{r}} \neq \varepsilon$ and from (5) and (6) we know that $i_\mathtt{r} > 2 * |\mathtt{split}(\mathtt{trace}(\alpha \cdot G))|$.

We identify the role in $\mathcal{S}$ with the *earliest disagreement* in ρ_c: let $\bar{\mathtt{r}}$ be the role with the smallest $i_{\bar{\mathtt{r}}}$ in $\mathcal{S}$. Let $y_{\bar{\mathtt{r}}}$ denote $\mathtt{split}(\mathtt{trace}(\rho_c[i_{\bar{\mathtt{r}}}]))\Downarrow_{\Sigma_{\bar{\mathtt{r}}}}$.

Claim. $y_{\bar{\mathtt{r}}}$ must be a send event.

Assume by contradiction that $y_{\bar{\mathtt{r}}}$ is a receive event. We identify the symbol in w that disagrees with $y_{\bar{\mathtt{r}}}$: let w' be the largest prefix of w such that $w'\Downarrow_{\Sigma_{\bar{\mathtt{r}}}} \leq \mathtt{split}(\mathtt{trace}(\rho_c))$. By definition, $w'\Downarrow_{\Sigma_{\bar{\mathtt{r}}}} = \mathtt{split}(\mathtt{trace}(\rho_c[0..i_{\bar{\mathtt{r}}}-1]))\Downarrow_{\Sigma_{\bar{\mathtt{r}}}}$. Let z be the next symbol following w' in w; then $z \in \Sigma_{\bar{\mathtt{r}}}$ and $z \neq y_{\bar{\mathtt{r}}}$. Furthermore, by No Mixed Choice (5.5) we have that $z \in \Sigma_?$.

By assumption, $w'z \not\leq \mathtt{split}(\mathtt{trace}(\rho_c[0..i_{\bar{\mathtt{r}}}]))$. Therefore, any run that begins with $\rho_c[0..i_{\bar{\mathtt{r}}}]$ cannot be contained in $\mathrm{R}^{\mathbf{G}}_{\bar{\mathtt{r}}}(w'z)$, or consequently in $I(w'z)$. We show however, that $I(w'z)$ must contain some runs that begin with $\rho_c[0..i_{\bar{\mathtt{r}}}]$. From Lemma 6.4 for traces w' and $w'z$, we obtain that $I(w') = I(w'z)$. Therefore, it suffices to show that $I(w')$ contains runs that begin with $\rho_c[0..i_{\bar{\mathtt{r}}}]$.

Claim $\forall w'' \leq w'.\, I(w'')$ contains runs that begin with $\rho_c[0..i_{\bar{\mathtt{r}}}]$.

We prove the claim via induction on w'.

The base case is trivial from the fact that $I(\varepsilon)$ contains all maximal runs.

For the inductive step, let $w''y \leq w'$.

In the case that $y \in \Sigma_?$, from Lemma 6.4 $I(w''y) = I(w'')$ and the witness from $I(w'')$ can be reused.

In the case that $y \in \Sigma_!$, let $\mathtt{s}$ be the active role of y and let ρ' be a run in $I(w'')$ beginning with $\rho_c[0..i_{\bar{\mathtt{r}}}]$ given by the inner induction hypothesis. Let $\alpha' \cdot G' \xrightarrow{l'} G'' \cdot \beta'$ be the unique splitting of ρ' for $\mathtt{s}$ with respect to w''. If

$\texttt{split}(l')\Downarrow_{\Sigma_\texttt{s}} = y$, then ρ' can be used as the witness. Otherwise, $\texttt{split}(l')\Downarrow_{\Sigma_\texttt{s}} \neq y$, and $\rho' \notin \mathrm{R}^{\mathbf{G}}_{\texttt{s}}(w''y)$.

The outer induction hypothesis holds for all prefixes of w: we instantiate it with w'' and y to obtain:

$$\exists\ \rho'' \in I(w''y).\ \alpha' \cdot G' \leq \rho''\ \ .$$

Let $i_\texttt{s}$ be defined as before; it follows that $\rho'[i_\texttt{s}] = G'$. It must be the case that $i_\texttt{s} > i_{\bar{\texttt{r}}}$: if $i_\texttt{s} \leq i_{\bar{\texttt{r}}}$, because ρ_c and ρ' share a prefix $\rho_c[0..i_{\bar{\texttt{r}}}]$ and $w''y \leq w$, $\texttt{s}$ would be the earliest disagreeing role instead of $\bar{\texttt{r}}$.

Because $i_\texttt{s} > i_{\bar{\texttt{r}}}$, $\rho_c[0..i_{\bar{\texttt{r}}}] = \rho'[0..i_{\bar{\texttt{r}}}] \leq \rho'[0..i_\texttt{s}]$. Because $\rho'[0..i_\texttt{s}] = \alpha' \cdot G' \leq \rho''$, it follows from prefix transitivity that $\rho_c[0..i_{\bar{\texttt{r}}}] \leq \rho''$, thus re-establishing the induction hypothesis for $w''y$ with ρ'' as a witness run that begins with $\rho_c[0..i_{\bar{\texttt{r}}}]$.

This concludes our proof that $I(w')$ contains runs that begin with $\rho_c[0..i_{\bar{\texttt{r}}}]$, and in turn our proof by contradiction that $y_{\bar{\texttt{r}}}$ must be a receive event.

We can rewrite candidate run ρ_c as follows:

$$\rho_c = G_0 \xrightarrow{l_0} G_1 \ldots G_{i_{\bar{\texttt{r}}}} \xrightarrow{l_{i_{\bar{\texttt{r}}}}} G_{i_{\bar{\texttt{r}}}+1} \ldots\ \ .$$

We have established that $l_{i_{\bar{\texttt{r}}}}$ must be a send event for $\bar{\texttt{r}}$. We can reason from Send Validity similarly to our construction of $\bar{\rho}$'s prefix above, and conclude that there exists a transition label and maximal suffix from $G_{i_{\bar{\texttt{r}}}}$ such that the resulting run no longer disagrees with $w\Downarrow_{\Sigma_{\bar{\texttt{r}}}}$. We update our candidate run ρ_c with the correct transition label and maximal suffix, update the set of states $\mathcal{S} \in \mathcal{P}$ to the new set of roles that disagree with the new candidate run, and repeat the construction above on the new candidate run until $\mathcal{S}$ is empty.

Termination is guaranteed in at most $|w|$ steps by the fact that the number of symbols in w that agree with the candidate run up to $i_{\bar{\texttt{r}}}$ must increase.

Upon termination, the resulting $\bar{\rho}$ satisfies the final remaining property 3: $\bar{\rho} \in I$. This concludes the proof of the inductive step, and consequently the proof of the prefix-preservation of send transitions. ☐

Lemma 6.3. *Let* $\mathbf{G}$ *be a global type and* $\{\!\{\mathscr{P}(\mathbf{G}, \texttt{p})\}\!\}_{\texttt{p}\in\mathcal{P}}$ *be the subset projection. Let* w *be a trace of* $\{\!\{\mathscr{P}(\mathbf{G}, \texttt{p})\}\!\}_{\texttt{p}\in\mathcal{P}}$. *It holds that* $I(w)$ *is non-empty.*

Proof. We prove the claim by induction on the length of w.

Base Case. $w = \varepsilon$. The trace $w = \varepsilon$ is trivially consistent with all maximal runs, and $I(w)$ therefore contains all maximal runs. By definition of $\mathbf{G}$, language $\mathcal{L}(\mathbf{G})$ is non-empty and at least one maximal run exists. Thus, $I(w)$ is non-empty.

Induction Step. Let wx be an extension of w by $x \in \Sigma_{async}$.

The induction hypothesis states that $I(w) \neq \emptyset$. To re-establish the induction hypothesis, we need to show $I(wx) \neq \emptyset$. We proceed by case analysis on whether x is a receive or send event.

Receive Case. Let $x = \texttt{p}\triangleleft\texttt{q}?m$. By Lemma 6.4, $I(wx) = I(w)$. $I(wx) \neq \emptyset$ follows trivially from the induction hypothesis and this equality.

Send Case. Let $x = \mathtt{p} \triangleright \mathtt{q}!m$. By Lemma 6.6, there exists a run in $I(wx)$ that shares a prefix with a run in $I(w)$. $I(wx) \neq \emptyset$ again follows trivially. ☐

Lemma C.7. *Let* $\mathbf{G}$ *be a global type and* $\{\!\{\mathscr{C}(\mathbf{G}, \mathtt{p})\}\!\}_{\mathtt{p} \in \mathcal{P}}$ *be the subset construction. Let* w *be a trace of* $\{\!\{\mathscr{C}(\mathbf{G}, \mathtt{p})\}\!\}_{\mathtt{p} \in \mathcal{P}}$*. If* w *is terminated, then* w *is* $\mathbf{G}$*-complete.*

Proof. We prove the claim by contraposition and assume that w is not $\mathbf{G}$-complete. Then, there exists a run $\rho \in I(w)$ and a non-empty set of roles $\mathcal{S}$ such that for every $\mathtt{r} \in \mathcal{S}$, it holds that $w{\Downarrow_{\Sigma_\mathtt{r}}} \neq \big(\mathtt{split}(\mathtt{trace}(\rho))\big){\Downarrow_{\Sigma_\mathtt{r}}}$ (*). Since w is a trace, we know there exists a run $(\vec{s}_0, \xi_0) \xrightarrow{w_0} \ldots \xrightarrow{w_{n-1}} (\vec{s}_n, \xi_n)$ of $\{\!\{\mathscr{C}(\mathbf{G}, \mathtt{p})\}\!\}_{\mathtt{p} \in \mathcal{P}}$ such that $w = w_0 \ldots w_{n-1}$. We need to show that there exists $(\vec{s}_{n+1}, \xi_{n+1})$ with $(\vec{s}_n, \xi_n) \xrightarrow{w_n} (\vec{s}_{n+1}, \xi_{n+1})$ for some w_n. Given some role $\mathtt{p}$, let $\rho_\mathtt{p}$ denote the largest prefix of ρ that contains $\mathtt{p}$'s local view of w. Formally,

$$\rho_\mathtt{p} = max\{\rho' \mid \rho' \leq \rho \ \wedge \ \mathtt{split}(\mathtt{trace}(\rho')){\Downarrow_{\Sigma_\mathtt{p}}} = w{\Downarrow_{\Sigma_\mathtt{p}}}\} \ .$$

Note that due to maximality, the next transition in ρ after $\rho_\mathtt{p}$ must have $\mathtt{p}$ as its active role. Let $\mathtt{q}$ be the role in $\mathcal{S}$ for whom $\rho_\mathtt{q}$ is the smallest. From Lemma 4.3 and (*), it follows that $\vec{s}_{n,\mathtt{q}}$ has outgoing transitions. If $q_{n,\mathtt{q}}$ has outgoing send transitions, then (q_{n+1}, ξ_{n+1}) exists trivially. If $q_{n,\mathtt{q}}$ has outgoing receive transitions, it must be the case that the next transition in ρ after $\rho_\mathtt{q}$ is of the form $\mathtt{p} \to \mathtt{q} : m$ for some $\mathtt{p}$ and m. From the fact that $\mathtt{q}$ is the role with the smallest $\rho_\mathtt{q}$, we know that $\rho_\mathtt{q} < \rho_\mathtt{p}$, and from the FIFO property of CSM channels it follows that m is in $\xi_n(\mathtt{p}, \mathtt{q})$. Then, the receive transition is enabled for $\mathtt{q}$, and there exists $(\vec{s}_{n+1}, \xi_{n+1})$ with $(\vec{s}_n, \xi_n) \xrightarrow{\mathtt{p} \triangleleft \mathtt{p}?m} (\vec{s}_{n+1}, \xi_{n+1})$. This shows that $w = w_1 \ldots w_{n-1}$ is not terminated and concludes the proof. ☐

Lemma C.8. *Let* $\mathbf{G}$ *be a global type and* $\{\!\{\mathscr{P}(\mathbf{G}, \mathtt{p})\}\!\}_{\mathtt{p} \in \mathcal{P}}$ *be the subset projection. Let* w *be a trace of* $\{\!\{\mathscr{P}(\mathbf{G}, \mathtt{p})\}\!\}_{\mathtt{p} \in \mathcal{P}}$*. If* w *is* $\mathbf{G}$*-complete, then* $w \in \mathcal{L}(\mathbf{G})$*.*

Proof. By definition of w being $\mathbf{G}$-complete,

$$\forall \mathtt{p} \in \mathcal{P}, \ \rho \in I(w). \ w{\Downarrow_{\Sigma_\mathtt{p}}} = \big(\mathtt{split}(\mathtt{trace}(\rho))\big){\Downarrow_{\Sigma_\mathtt{p}}} \ .$$

From Lemma 6.3, $I(w)$ is non-empty. Let ρ be a run in $I(w)$, and let $w' = \mathtt{split}(\mathtt{trace}(\rho)) \in \mathcal{L}(\mathbf{G})$. By the semantics of $\mathcal{L}(\mathbf{G})$, $\mathcal{L}(\mathbf{G})$ is closed under the $\sim$ relation, and thus it suffices to show that $w \sim w'$. [31, Lemma 23] states that if w is channel-compliant [31, Definition 19], then $w \sim w'$ iff w' is channel-compliant and forall $\mathtt{p} \in \mathcal{P}$, $w{\Downarrow_{\Sigma_\mathtt{p}}} = w'{\Downarrow_{\Sigma_\mathtt{p}}}$. The fact that w is channel-compliant follows from [31, Lemma 20] and w being a a CSM trace; w' is channel-compliant by construction, and the last condition is satisfied by assumption that w is $\mathbf{G}$-complete and by definition of w'. Thus, we conclude that $w \sim w'$. ☐

Theorem 6.1. *Let* $\mathbf{G}$ *be a global type and* $\{\!\{\mathscr{P}(\mathbf{G}, \mathtt{p})\}\!\}_{\mathtt{p} \in \mathcal{P}}$ *be the subset projection. Then,* $\{\!\{\mathscr{P}(\mathbf{G}, \mathtt{p})\}\!\}_{\mathtt{p} \in \mathcal{P}}$ *implements* $\mathbf{G}$*.*

Proof. First, we show that $\{\!\{\mathscr{P}(\mathbf{G},\mathtt{p})\}\!\}_{\mathtt{p}\in\mathcal{P}}$ is deadlock-free, namely, that every finite trace extends to a maximal trace. Let w be a trace of $\{\!\{\mathscr{P}(\mathbf{G},\mathtt{p})\}\!\}_{\mathtt{p}\in\mathcal{P}}$. Let w' denote the extension of w. If $w' \in \Sigma^\omega_{async}$, then w' is maximal and we are done. Otherwise, we have $w' \in \Sigma^*_{async}$. Let $(\vec{s}', \xi')$ denote the $\{\!\{\mathscr{P}(\mathbf{G},\mathtt{p})\}\!\}_{\mathtt{p}\in\mathcal{P}}$ configuration reached on w'. By definition of w' being the largest extension, w' is a terminated trace, and there exists no configuration reachable from $(\vec{s}', \xi')$. By Lemma C.7, w' is $\mathbf{G}$-complete. By Lemma C.8, $w' \in \mathcal{L}(\mathbf{G})$. Therefore, all states in $\vec{s}'$ are final and all channels in ξ are empty, and w' is a maximal trace in $\{\!\{\mathscr{P}(\mathbf{G},\mathtt{p})\}\!\}_{\mathtt{p}\in\mathcal{P}}$.

This concludes our proof that $\{\!\{\mathscr{P}(\mathbf{G},\mathtt{p})\}\!\}_{\mathtt{p}\in\mathcal{P}}$ is deadlock-free.

Next, we show that $\mathcal{L}(\{\!\{\mathscr{P}(\mathbf{G},\mathtt{p})\}\!\}_{\mathtt{p}\in\mathcal{P}}) = \mathcal{L}(\mathbf{G})$. The backward direction, $\mathcal{L}(\mathbf{G}) \subseteq \mathcal{L}(\{\!\{\mathscr{P}(\mathbf{G},\mathtt{p})\}\!\}_{\mathtt{p}\in\mathcal{P}})$, is given by Lemma 4.4. For the forward direction, let $w \in \mathcal{L}(\{\!\{\mathscr{P}(\mathbf{G},\mathtt{p})\}\!\}_{\mathtt{p}\in\mathcal{P}})$, and let $(\vec{s}, \xi)$ denote the configuration reached on w. We proceed by case analysis on whether w is a finite or infinite maximal trace.

Case: $w \in \Sigma^*_{async}$. We show a stronger property: w is a terminated trace. Then, we use Lemma C.7 and Lemma C.8 as above to obtain $w \in \mathcal{L}(\mathbf{G})$. By definition of $(\vec{s}, \xi)$ being final, all states in $\vec{s}$ are final and all channels in ξ are empty. We argue there does not exist a configuration reachable from $(\vec{s}, \xi)$. From Proposition C.4, all outgoing states from states in $\vec{s}$ must be receive transitions. However, no receive transitions are enabled because all channels in ξ are empty. Therefore, $(\vec{s}, \xi)$ is a terminated configuration and w is a terminated trace.

Case: $w \in \Sigma^\omega$. By the semantics of $\mathcal{L}(\mathbf{G})$, to show $w \in \mathcal{L}(\mathbf{G})$ it suffices to show:

$$\exists w' \in \Sigma^\omega.\ w' \in \mathtt{split}(\mathcal{L}(\mathsf{GAut}(\mathbf{G}))) \wedge w \preceq^\omega_\sim w' \ .$$

Claim. $\bigcap_{u \le w} I(u)$ contains an infinite run.

First, we show that there exists an infinite run in $\mathsf{GAut}(\mathbf{G})$. We apply König's Lemma to an infinite tree where each vertex corresponds to a finite run. We obtain the vertex set from the intersection sets of w's prefixes; each prefix "contributes" a set of finite runs. Formally, for each prefix $u \le w$, let V_u be defined as:

$$V_u := \bigcup_{\rho_u \in I(u)} \min\{\rho' \mid \rho' \le \rho_u \wedge \forall \mathtt{p} \in \mathcal{P}.\ u{\Downarrow_{\Sigma_\mathtt{p}}} \le \mathtt{split}(\mathtt{trace}(\rho')){\Downarrow_{\Sigma_\mathtt{p}}}\} \ .$$

By Lemma 6.3, V_u is guaranteed to be non-empty. We construct a tree $\mathcal{T}_w(V, E)$ with $V := \bigcup_{u \le w} V_u$ and $E := \{(\rho_1, \rho_2) \mid \rho_1 \le \rho_2\}$. The tree is rooted in the empty run, which is included V by V_ε. V is infinite because there are infinitely many prefixes of w. $\mathcal{T}_w$ is finitely branching due to the finiteness of $\delta_\mathbf{G}$ and the fact that each vertex represents a finite run. Therefore, there must exist a ray in $\mathcal{T}_w$ representing an infinite run in $\mathsf{GAut}(\mathbf{G})$.

Let ρ' be such an infinite run. We now show that $\rho' \in \bigcap_{u \le w} I(u)$. Let v be a prefix of w. To show that $\rho' \in I(v)$, it suffices to show that one of the vertices in V_v lies on ρ'. In other words,

$$V_v \cap \{v \mid v \in \rho'\} \ne \emptyset \ .$$

Assume by contradiction that ρ' passes through none of the vertices in V_v. Then, for any $u' \geq u$, because intersection sets are monotonically decreasing, it must be the case that ρ' passes through none of the vertices in V'_u. Therefore, ρ' can only pass through vertices in V''_u, where $u'' \leq u$. However, the set $\bigcup_{u'' \leq u} V''_u$ has finite cardinality. We reach a contradiction, concluding our proof of the above claim.

Let $\rho' \in \bigcap_{u \leq w} I(u)$, and let $w' = \mathtt{split}(\mathtt{trace}(\rho'))$. It is clear that $w' \in \Sigma^{\omega}_{async}$ and $w' \in \mathtt{split}(\mathcal{L}(\mathsf{GAut}(\mathbf{G})))$. It remains to show that $w \preceq^{\omega}_{\sim} w'$. By the definition of $\preceq^{\omega}_{\sim}$, it further suffices to show that:

$$\forall u \leq w,\ \exists u' \leq w', v \in \Sigma^*.\ uv \sim u' \ \ .$$

Let u be an arbitrary prefix of w. Because by definition $\rho' \in I(u)$, it holds that $u{\Downarrow}_{\Sigma_{\mathtt{p}}} \leq \mathtt{split}(\mathtt{trace}(\rho')){\Downarrow}_{\Sigma_{\mathtt{p}}}$.

For each role $\mathtt{p} \in \mathcal{P}$, let $\rho'_{\mathtt{p}}$ be defined as the largest prefix of ρ' such that $\mathtt{split}(\mathtt{trace}(\rho'_{\mathtt{p}})){\Downarrow}_{\Sigma_{\mathtt{p}}} = u{\Downarrow}_{\Sigma_{\mathtt{p}}}$. Such a run is well-defined by the fact that u is a prefix of an infinite word w, and there exists a longer prefix v such that $u \leq v$ and $v{\Downarrow}_{\Sigma_{\mathtt{p}}} \leq \mathtt{split}(\mathtt{trace}(\rho')){\Downarrow}_{\Sigma_{\mathtt{p}}}$.

Let $\mathtt{s}$ be the role with the maximum $|\rho'_{\mathtt{s}}|$ in $\mathcal{P}$. Let $u' = \mathtt{split}(\mathtt{trace}(\rho'_{\mathtt{s}}))$. Clearly, $u' \leq w'$. Because u' is $\mathtt{split}(\mathtt{trace}(\rho'_{\mathtt{s}}))$ for the role with the longest $\rho'_{\mathtt{s}}$, it holds for all roles $\mathtt{p} \in \mathcal{P}$ that $u{\Downarrow}_{\Sigma_{\mathtt{p}}} \leq u'{\Downarrow}_{\Sigma_{\mathtt{p}}}$. Then, there must exist $y_{\mathtt{p}} \in {\Sigma_{\mathtt{p}}}^*$ such that

$$u{\Downarrow}_{\Sigma_{\mathtt{p}}} \cdot y_{\mathtt{p}} = u'{\Downarrow}_{\Sigma_{\mathtt{p}}} \ \ .$$

Let $y_{\mathtt{p}}$ be defined in this way for each role. We construct $v \in \Sigma^*$ such that $uv \sim u'$. Let v be initialized with ε. If there exists some role in $\mathcal{P}$ such that $y_{\mathtt{p}}[0] \in \Sigma_{\mathtt{p},!}$, append $y_{\mathtt{p}}$ to v and update $y_{\mathtt{p}}$. If not, for all roles $\mathtt{p} \in \mathcal{P}$, $y_{\mathtt{p}}[0] \in \Sigma_{\mathtt{p},?}$. Each symbol $y_{\mathtt{p}}[0]$ for all roles appears in u'. Let $i_{\mathtt{p}}$ denote for each role the index in u' such that $u'[i] = y_{\mathtt{p}}[0]$. Let $\mathtt{r}$ be the role with the minimum index $i_{\mathtt{r}}$. Append $y_{\mathtt{r}}$ to v and update $y_{\mathtt{r}}$. Termination is guaranteed by the strictly decreasing measure of $\sum_{\mathtt{p} \in \mathcal{P}} |y_{\mathtt{p}}|$.

We argue that uv satisfies the inductive invariant of channel compliancy. In the case where v is extended with a send action, channel compliancy is trivially re-established. In the receive case, channel compliancy is re-established by the fact that the append order for receive actions follows that in u', which is channel-compliant by construction. We conclude that $uv \sim u'$ by applying [31, Lemma 22]. $\square$

D Additional Material for §7

Lemma D.1. *Let* $\mathtt{p}$ *be a role,* $\mathbf{G}$ *be a global type,* G' *be a syntactic subterm of* $\mathbf{G}$*, and* $\{\!\!\{\mathscr{C}(\mathbf{G}, \mathtt{p})\}\!\!\}_{\mathtt{p} \in \mathcal{P}}$ *be its subset construction. Let* s *be some state in* $Q_{\mathtt{p}}$ *with* $G' \in s$*. Then, there is a run* $\rho_{\mathbf{G}}$ *in* $\mathsf{GAut}(\mathbf{G})$ *ending in state* q'_G*, i.e.*

$$\rho_{\mathbf{G}} = q_{0,\mathbf{G}} \xrightarrow{trace(\rho_{\mathbf{G}})}{}^{*}\ q'_G,$$

such that $\mathscr{C}(\mathbf{G}, \mathtt{p})$ will reach s on the projected trace, i.e.,

$$\rho_{\mathtt{p}} = s_{0,\mathtt{p}} \xrightarrow{\mathtt{split}(\mathtt{trace}(\rho_{\mathbf{G}}))\Downarrow_{\Sigma_{\mathtt{p}}}}{}^{*} s.$$

Proof. Recall that the set of states for $\mathscr{C}(\mathbf{G}, \mathtt{p})$ is defined as a least fixed point:

$$Q_{\mathtt{p}} := \mathrm{lfp}^{\subseteq}_{\{s_{0,\mathtt{p}}\}} \lambda Q.\, Q \cup \{\delta(s, a) \mid s \in Q \wedge a \in \Sigma_{\mathtt{p}}\} \setminus \{\emptyset\}$$

where $\delta(s, a)$ is an intermediate transition relation that is defined for all subsets $s \subseteq Q_{\mathbf{G}}$ and every event $a \in \Sigma_{\mathtt{p}}$ as follows:

$$\delta(s, a) := \{q' \in Q_{\mathbf{G}} \mid \exists q \in s, q \xrightarrow{a}\xrightarrow{\varepsilon}{}^{*} q' \in \delta_{\downarrow}\}$$

From the definition of $Q_{\mathtt{p}}$, there exists a sequence of states $s_1, \ldots, s_n$ such that $s_1 = s_{0,\mathtt{p}}$, $s_n = s$ and for every $i \in \{1, \ldots, n-1\}$, it holds that

$$\exists a \in \Sigma_{\mathtt{p}}.\ \delta(s_i, a) = s_{i+1}$$

Let a_i denote the existential witness for each i. From the definition of $\delta(s, a)$, for every $i \in \{1, \ldots, n-1\}$, it follows that

$$\forall q' \in s_{i+1}.\ \exists q \in s_i.\ q \xrightarrow{a_i}\xrightarrow{\varepsilon}{}^{*} q' \in \delta_{\downarrow}$$

By assumption, $G' \in s$. There then exists a sequence of global syntactic subterms $G_1, \ldots, G_n$ such that $G_1 = \mathbf{G}$, $G_n = G'$ and for every $i \in \{1, \ldots, n-1\}$, it holds that

$$G_i \in s_i \wedge G_{i+1} \in s_{i+1} \wedge G_i \xrightarrow{a_i}\xrightarrow{\varepsilon}{}^{*} G_{i+1} \in \delta_{\downarrow}$$

We can expand ε^*: for every $i \in \{1, \ldots, n-1\}$, there exists $k_i \geq 0$ and a sequence of syntactic subterms $G_{i,0}, \ldots, G_{i,k_i}$ such that $G_{i,0} = G_i$ and $G_{i,k_i} = G_{i+1}$ and

$$G_{i,0} \xrightarrow{a} G_{i,1} \in \delta_{\downarrow} \text{ and } G_{i,j} \xrightarrow{\varepsilon} G_{i,j+1} \in \delta_{\downarrow} \text{ for every } j \in \{1, k_i - 1\}.$$

This expansion yields a run $\rho_{\downarrow}$ in the projection by erasure $\mathsf{GAut}(\mathbf{G})\downarrow_{\mathtt{p}}$. Because of recursion terms, the expansion might not be unique, but we can pick the smallest k_i possible for every i. With the definition of $\delta_{\downarrow}$, it is trivial to translate this run in $\mathsf{GAut}(\mathbf{G})\downarrow_{\mathtt{p}}$ to a run $\rho_{\mathbf{G}}$ in $\mathsf{GAut}(\mathbf{G})$: the events $a \in \Sigma_{\mathtt{p}}$ become $a' \in \Sigma_{sync}$ such that $\mathtt{split}(a')\Downarrow_{\Sigma_{\mathtt{p}}} = a$ and ε becomes $b \in \Sigma_{sync}$ such that $\mathtt{split}(b)\Downarrow_{\Sigma_{\mathtt{p}}} = \varepsilon$.

It is clear by construction that $s_1, \ldots, s_n$ (with its corresponding transitions) serves as a witness for $\rho_{\mathtt{p}}$, while $G_{1,0}, \ldots, G_{1,k_1}, \ldots, G_n$ (with its respective transitions) serves as a witness for $\rho_{\mathbf{G}}$. □

Lemma D.2. *Let $\mathbf{G}$ be a global type and let $\{\!\{B_{\mathtt{p}}\}\!\}_{\mathtt{p} \in \mathcal{P}}$ implement $\mathbf{G}$ with $B_{\mathtt{p}} = (Q_{B,\mathtt{p}}, \delta_{B,\mathtt{p}}, s_{B,0,\mathtt{p}}, F_{B,0,\mathtt{p}})$ for role $\mathtt{p}$. Let $s \in Q_{\mathtt{p}}$, $x \in \Sigma_{\mathtt{p}}$ and $s \xrightarrow{x} t \in \delta_{\mathtt{p}}$ from the subset construction $\mathscr{C}(\mathbf{G}, \mathtt{p})$. Let $u \in \Sigma_{\mathtt{p}}{}^{*}$ such that $s_{0,\mathtt{p}} \xrightarrow{u}{}^{*} s$. Then, there exists $s', t' \in Q_{B,\mathtt{p}}$ such that $s_{B,0,\mathtt{p}} \xrightarrow{u}{}^{*} s'$ and $s' \xrightarrow{x} t' \in \delta_{B,\mathtt{p}}$.*

Proof. Because $\{\!\{B_{\mathtt{p}}\}\!\}_{\mathtt{p}\in\mathcal{P}}$ implements $\mathbf{G}$, it must hold that $\mathcal{L}(\mathbf{G})\Downarrow_{\Sigma_{\mathtt{p}}} \subseteq \mathcal{L}(B_{\mathtt{p}})$ since $B_{\mathtt{p}}$ must produce at least the behaviors as specified by $\mathbf{G}$ for its role. It follows that $\text{pref}(\mathcal{L}(\mathbf{G})\Downarrow_{\Sigma_{\mathtt{p}}}) \subseteq \text{pref}(\mathcal{L}(B_{\mathtt{p}}))$. From Lemma 4.3, we know that $\mathcal{L}(\mathbf{G})\Downarrow_{\Sigma_{\mathtt{p}}} = \mathscr{C}(\mathbf{G}, \mathtt{p})$. By construction of $\mathscr{C}(\mathbf{G}, \mathtt{p})$, if ux is reachable from the initial state in $\mathscr{C}(\mathbf{G}, \mathtt{p})$ then ux is the prefix of some word in $\mathcal{L}(\mathbf{G})\Downarrow_{\Sigma_{\mathtt{p}}}$. Therefore, it holds that $ux \in \text{pref}(\mathcal{L}(\mathbf{G})\Downarrow_{\Sigma_{\mathtt{p}}})$ and consequently $ux \in \text{pref}(\mathcal{L}(B_{\mathtt{p}}))$. Because $B_{\mathtt{p}}$ is deterministic, there exists a unique t' such that $B_{\mathtt{p}}$ reaches t' from the initial state on ux. This concludes our proof.

Theorem 7.1 (Completeness). *If* $\mathbf{G}$ *is implementable, then* $\{\!\{\mathscr{P}(\mathbf{G}, \mathtt{p})\}\!\}_{\mathtt{p}\in\mathcal{P}}$ *is defined.*

Proof. From the fact that $\mathbf{G}$ is implementable, we know there exists a CSM $\{\!\{B_{\mathtt{p}}\}\!\}_{\mathtt{p}\in\mathcal{P}}$ that implements $\mathbf{G}$. Showing that the subset projection is defined amounts to showing that Send and Receive Validity (Definitions 5.2 and 5.3) hold for the subset construction.

We proceed by contradiction and assume the negation of Send and Receive Validity in turn, and in each case derive a contradiction to the fact that $\{\!\{B_{\mathtt{p}}\}\!\}_{\mathtt{p}\in\mathcal{P}}$ implements $\mathbf{G}$. Specifically, we contradict protocol fidelity (Definition 3.1(i)), and show that $\mathcal{L}(\mathbf{G}) \neq \mathcal{L}(\{\!\{B_{\mathtt{p}}\}\!\}_{\mathtt{p}\in\mathcal{P}})$.

To prove the inequality of the two languages, it suffices to prove the inequality of their respective prefix sets, i.e.

$$\{u \mid u \leq w \wedge w \in \mathcal{L}(\mathbf{G})\} \neq \{u \mid u \leq w \wedge w \in \mathcal{L}(\{\!\{B_{\mathtt{p}}\}\!\}_{\mathtt{p}\in\mathcal{P}})\}$$

Specifically, we show there is $v \in \Sigma^*_{async}$ such that

$$\begin{aligned} &v \in \{u \mid u \leq w \wedge w \in \mathcal{L}(\{\!\{B_{\mathtt{p}}\}\!\}_{\mathtt{p}\in\mathcal{P}})\} \wedge \\ &v \notin \{u \mid u \leq w \wedge w \in \mathcal{L}(\mathbf{G})\}\ . \end{aligned}$$

Because $\{\!\{B_{\mathtt{p}}\}\!\}_{\mathtt{p}\in\mathcal{P}}$ is deadlock-free by assumption, every trace either can be extended to end in a final configuration or to be infinite. Therefore, any word $v \in \Sigma^*_{async}$ that is a trace of $\{\!\{B_{\mathtt{p}}\}\!\}_{\mathtt{p}\in\mathcal{P}}$ is a member of the prefix set, i.e.

$$\exists\ (\vec{s}, \xi).\ (\vec{s}_0, \xi_0) \xrightarrow{v}{}^{*} (\vec{s}, \xi) \implies v \in \{u \mid u \leq w \wedge w \in \mathcal{L}(\{\!\{B_{\mathtt{p}}\}\!\}_{\mathtt{p}\in\mathcal{P}})\}\ .$$

By the semantics of $\mathcal{L}(\mathbf{G})$, for any $w \in \mathcal{L}(\mathbf{G})$, there exists $w' \in \mathcal{L}(\mathsf{GAut}(\mathbf{G}))$ with $w \sim \mathtt{split}(w')$. For any $w' \in \mathcal{L}(\mathsf{GAut}(\mathbf{G}))$, it is straightforward that $I(\mathtt{split}(w')) \neq \emptyset$. Because intersection sets are closed under the indistinguishability relation (Corollary C.1), it holds that $I(w) \neq \emptyset$. Because $I(\text{-})$ is monotonically decreasing, if $I(w)$ is non-empty then for any $v \leq w$, $I(v)$ is non-empty. By the following, to show that a word v is not a member of the prefix set of $\mathcal{L}(\mathbf{G})$ it suffices to show that $I(v)$ is empty:

$$\forall v \in \Sigma^*_{async}.\ I(v) = \emptyset \implies \forall w.\ v \leq w \implies w \notin \mathcal{L}(\mathbf{G})\ .$$

Therefore, under the assumption of the negation of Send or Receive Validity respectively, we explicitly construct a witness v_0 satisfying:

(a) v_0 is a trace of $\{\!\{B_{\mathtt{p}}\}\!\}_{\mathtt{p}\in\mathcal{P}}$, and
(b) $I(v_0) = \emptyset$.

Send Validity (Definition 5.2). Assume that Send Validity does not hold for some role $\mathtt{p} \in \mathcal{P}$. Let $s \in Q_{\mathtt{p}}$ be a state and $s \xrightarrow{\mathtt{p}\triangleright\mathtt{q}!m} s' \in \delta_{\mathtt{p}}$ a transition in the subset construction $\mathscr{C}(\mathbf{G}, \mathtt{p})$ such that

$$\text{tr-orig}(s \xrightarrow{\mathtt{p}\triangleright\mathtt{q}!m} s') \neq s \ .$$

Let D denote $s \setminus \text{tr-orig}(s \xrightarrow{\mathtt{p}\triangleright\mathtt{q}!m} s')$. By the negation of Send Validity, D is non-empty. Let G' be a syntactic subterm in D.

Because $G' \in s$, it follows from Lemma D.1 that there exists α such that $\alpha \cdot G'$ is a run in $\mathsf{GAut}(\mathbf{G})$. Let $\bar{w}$ be $\mathtt{split}(\mathtt{trace}(\alpha \cdot G'))$. Because $\{\!\{B_{\mathtt{p}}\}\!\}_{\mathtt{p}\in\mathcal{P}}$ implements $\mathbf{G}$, there exists a configuration $(\vec{t}, \xi)$ of $\{\!\{B_{\mathtt{p}}\}\!\}_{\mathtt{p}\in\mathcal{P}}$ such that $(\vec{t}_0, \xi_0) \xrightarrow{\bar{w}}{}^{*} (\vec{t}, \xi)$. Instantiating Lemma D.2 with s, $s \xrightarrow{\mathtt{p}\triangleright\mathtt{q}!m} s'$ and $\mathtt{split}(\mathtt{trace}(\alpha \cdot G'))\Downarrow_{\Sigma_{\mathtt{p}}}$, it follows that $\vec{t}_{\mathtt{p}}$ has an outgoing transition labeled $\mathtt{p} \triangleright \mathtt{q}!m$. Let $\vec{t}_{\mathtt{p}} \xrightarrow{\mathtt{p}\triangleright\mathtt{q}!m} t''$ be this transition.

The send transitions of any local machine in a CSM are always enabled. Formally, for all $w \in \Sigma_{async}^{*}$, $x \in \Sigma_{!}$, and $\mathtt{r} \in \mathcal{P}$, if w is a trace of $\{\!\{B_{\mathtt{p}}\}\!\}_{\mathtt{p}\in\mathcal{P}}$ and $\vec{t}_{w,\mathtt{r}} \xrightarrow{x} \vec{t}'_{w,\mathtt{r}} \in \delta_{\mathtt{r}}$, then wx is a trace of $\{\!\{B_{\mathtt{p}}\}\!\}_{\mathtt{p}\in\mathcal{P}}$. Instantiating this fact with $\bar{w}$ and $\vec{t}_{\mathtt{p}} \xrightarrow{\mathtt{p}\triangleright\mathtt{q}!m} t''$, we obtain that $\bar{w} \cdot \mathtt{p} \triangleright \mathtt{q}!m$ is a trace of $\{\!\{B_{\mathtt{p}}\}\!\}_{\mathtt{p}\in\mathcal{P}}$.

Let $\bar{w} \cdot \mathtt{p}\triangleright\mathtt{q}!m$ be our witness v_0; it then follows that v_0 satisfies (a). It remains to show that v_0 satisfies (b), namely $I(\bar{w} \cdot \mathtt{p} \triangleright \mathtt{q}!m) = \emptyset$.

Claim. All runs in $I(\bar{w})$ begin with $\alpha \cdot G'$.

Proof of Claim. Recall that $\bar{w}$ is defined as $\mathtt{split}(\mathtt{trace}(\alpha \cdot G'))$. Assume by contradiction that $\rho' \in I(\bar{w})$ and ρ' does not begin with $\alpha \cdot G'$. Due to the syntactic structure of global runs, the first divergence between two runs must correspond to a syntactic subterm of the form $\sum_{i\in I} \mathtt{p}' \to \mathtt{q}'_i : m'_i.G'_i$. Let $\mathtt{p}'$ be the sender in the first divergence between ρ' and $\alpha \cdot G'$, and let the two runs respectively contain the subterms G'_i and G'_j. Because ρ' is in $\mathrm{R}^{\mathbf{G}}_{\mathtt{p}'}(\bar{w})$, it holds that $\bar{w}\Downarrow_{\Sigma_{\mathtt{p}'}} \leq \mathtt{split}(\mathtt{trace}(\rho'))\Downarrow_{\Sigma_{\mathtt{p}'}}$. Because $\bar{w} = \mathtt{split}(\mathtt{trace}(\alpha \cdot G'))$, we can rewrite the inequality as $\mathtt{split}(\mathtt{trace}(\alpha \cdot G'))\Downarrow_{\Sigma_{\mathtt{p}'}} \leq \mathtt{split}(\mathtt{trace}(\rho'))\Downarrow_{\Sigma_{\mathtt{p}'}}$.

We know that $\mathtt{split}(\mathtt{trace}(\alpha \cdot G'))\Downarrow_{\Sigma_{\mathtt{p}'}}$ and $\mathtt{split}(\mathtt{trace}(\rho'))\Downarrow_{\Sigma_{\mathtt{p}'}}$ share a common prefix, followed by different send actions from $\mathtt{p}'$, i.e., they are respectively of the form $x' \cdot \mathtt{p}' \triangleright \mathtt{q}_j!m'_j \cdot y'$ and $x' \cdot \mathtt{p}' \triangleright \mathtt{q}_i!m'_i \cdot z'$. We arrive at a contradiction.

End Proof of Claim.

Recall that $G' \in D$ and $D = s \setminus \text{tr-orig}(s \xrightarrow{\mathtt{p}\triangleright\mathtt{q}!m} s')$. By the definition of tr-orig$(-)$ (Definition 4.2), there does not exist a global syntactic subterm G'' with $G' \xrightarrow{l'}{}^{*} G'' \in \delta_{\mathbf{G}}$ such that $l'\Downarrow_{\Sigma_{\mathtt{p}}} = \mathtt{p} \triangleright \mathtt{q}!m$. Therefore, there does not exist a maximal run in $\mathrm{R}^{\mathbf{G}}_{\mathtt{p}}(\bar{w} \cdot \mathtt{p} \triangleright \mathtt{q}!m)$, and $I(\bar{w} \cdot \mathtt{p} \triangleright \mathtt{q}!m) = \emptyset$ follows.

Our witness $v_0 = \bar{w} \cdot \mathtt{p}\triangleright\mathtt{q}!m$ thus satisfies both conditions (a) and (b) required for a contradiction. This concludes our proof that Send Validity is required to hold.

Receive Validity (Definition 5.3). Assume that Receive Validity does not hold for some role $\mathtt{p} \in \mathcal{P}$. In other words, there exists $s \in Q_{\mathtt{p}}$ with two transitions $s \xrightarrow{\mathtt{p}\triangleleft \mathtt{q}_1?m_1} s_1$, $s \xrightarrow{\mathtt{p}\triangleleft \mathtt{q}_2?m_2} s_2 \in \delta_{\mathtt{p}}$ and $G_2 \in \text{tr-dest}(s \xrightarrow{\mathtt{p}\triangleleft \mathtt{q}_2?m_2} s_2)$ such that

$$\mathtt{q}_1 \neq \mathtt{q}_2 \land \mathtt{q}_1 \triangleright \mathtt{p}!m_1 \in M^{\mathtt{p}}_{(G_2 \ldots)} \enspace .$$

Claim I. There exists $u \in \Sigma^*_{async}$ such that both $u \cdot \mathtt{p}\triangleleft \mathtt{q}_1?m_1$ and $u \cdot \mathtt{p}\triangleleft \mathtt{q}_2?m_2$ are traces of $\{\!\{B_{\mathtt{p}}\}\!\}_{\mathtt{p}\in\mathcal{P}}$.

Proof of Claim I. By the negation of Receive Validity, $G_2 \in \text{tr-dest}(s \xrightarrow{\mathtt{p}\triangleleft \mathtt{q}_2?m_2} s_2) \subseteq s_2$. From Lemma D.1 for s_2 and $G_2 \in s_2$, there exists ρ' such that ρ' ends in G_2 and is a run in $\mathsf{GAut}(\mathbf{G})$. Because $\mathtt{split}(\mathtt{trace}(\rho'))$ is a prefix in $\mathcal{L}(\mathbf{G})$ and by assumption $\{\!\{B_{\mathtt{p}}\}\!\}_{\mathtt{p}\in\mathcal{P}}$ implements $\mathbf{G}$, there exists a $\{\!\{B_{\mathtt{p}}\}\!\}_{\mathtt{p}\in\mathcal{P}}$ configuration $(\vec{t}, \xi)$ such that $(\vec{t_0}, \xi_0) \xrightarrow{\mathtt{split}(\mathtt{trace}(\rho'))}{}^{*} (\vec{t}, \xi)$. By the subset construction, it holds that $\mathscr{C}(\mathbf{G}, \mathtt{p})$ reaches s on $\mathtt{split}(\mathtt{trace}(\rho'))$. Instantiating Lemma D.2 twice with $s \xrightarrow{\mathtt{p}\triangleleft \mathtt{q}_1?m_1} s_1$, $s \xrightarrow{\mathtt{p}\triangleleft \mathtt{q}_2?m_2} s_2$ and $\mathtt{split}(\mathtt{trace}(\rho'))\Downarrow_{\Sigma_{\mathtt{p}}}$, we obtain $t_1 \xrightarrow{\mathtt{p}\triangleleft \mathtt{q}_1?m_1} t_1'$ and $t_2 \xrightarrow{\mathtt{p}\triangleleft \mathtt{q}_2?m_2} t_2'$. From the determinacy of $B_{\mathtt{p}}$, it holds that $t_1 = t_2$. Therefore, it holds that $\vec{t}_{\mathtt{p}} = t_1$ and there exist two outgoing transitions from $\vec{t}_{\mathtt{p}}$ labeled with $\mathtt{p} \triangleleft \mathtt{q}_1?m_1$ and $\mathtt{p} \triangleleft \mathtt{q}_2?m_2$.

From the fact that $s \xrightarrow{\mathtt{p}\triangleleft \mathtt{q}_2?m_2} s_2 \in \delta_{\mathtt{p}}$, there exist $G_1 \in s$ and $G_2' \in \text{tr-dest}(s \xrightarrow{\mathtt{p}\triangleleft \mathtt{q}_2?m_2} s_2) \subseteq s_2$ such that $G_1 \xrightarrow{\mathtt{q}_2 \to \mathtt{p}:m_2} G_2' \in \delta_{\mathbf{G}}$. Either $G_2 = G_2'$, or G_2 is reachable from G_2' via ε-transitions for $\mathtt{p}$. Without loss of generality, assume that $G_2 = G_2'$; if $G_2' \neq G_2$ then G_2' can also be picked as the witness from the definition of M. We rewrite ρ' as follows:

$$\rho' := \alpha \cdot G_1 \xrightarrow{\mathtt{q}_2 \to \mathtt{p}:m_2} G_2$$

From the negation of Receive Validity, we know that

$$\mathtt{q}_1 \triangleright \mathtt{p}!m_1 \in M^{\mathtt{p}}_{(G_2 \ldots)}$$

Then, there exists some suffix β such that the transition $\xrightarrow{\mathtt{q}_1 \to \mathtt{p}:m_1}$ occurs in β and $\alpha \cdot G_1 \xrightarrow{\mathtt{q}_2 \to \mathtt{p}:m_2} G_2 \cdot \beta$ is a maximal run. Let ρ denote this maximal run. Let $G_3 \xrightarrow{\mathtt{q}_1 \to \mathtt{p}:m_1} G_4$ be the earliest occurrence of $\xrightarrow{\mathtt{q}_1 \to \mathtt{p}:m_1}$ in β. We rewrite the suffix β in ρ to reflect the existence of G_3, G_4:

$$\rho := \alpha \cdot G_1 \xrightarrow{\mathtt{q}_2 \to \mathtt{p}:m_2} G_2 \cdot \beta_1 \cdot G_3 \xrightarrow{\mathtt{q}_1 \to \mathtt{p}:m_1} G_4 \cdot \beta_2$$

Note that β_1 does not contain any transitions of the form $\xrightarrow{\mathtt{q}_1 \to \mathtt{p}:m_1}$.

Let $\bar{w}$ denote $\mathtt{split}(\mathtt{trace}(\alpha))$, and $\bar{v}$ denote $\mathtt{split}(\mathtt{trace}(\beta_1))$. To produce a witness for u, we show that $\bar{w} \cdot \mathtt{q}_2 \triangleright \mathtt{p}!m_2 \cdot \mathtt{q}_1 \triangleright \mathtt{p}!m_1$ is a trace of $\{\!\{B_{\mathtt{p}}\}\!\}_{\mathtt{p}\in\mathcal{P}}$, and in the resulting CSM configuration $(\vec{s}', \xi')$, $\vec{s}'_{\mathtt{p}}$ has two outgoing transitions labeled $\mathtt{p} \triangleleft \mathtt{q}_1?m_1$ and $\mathtt{p} \triangleleft \mathtt{q}_2?m_2$. Moreover, we show that the channels $\xi'(\mathtt{q}_1, \mathtt{p})$ and $\xi'(\mathtt{q}_2, \mathtt{p})$ respectively contain the messages m_1 and m_2 at the head.

First, we show that $\bar{w} \cdot \mathtt{q}_2 \triangleright \mathtt{p}!m_2 \cdot \mathtt{q}_1 \triangleright \mathtt{p}!m_1$ is a trace of $\{\!\{B_\mathtt{p}\}\!\}_{\mathtt{p}\in\mathcal{P}}$.

By assumption that $\{\!\{B_\mathtt{p}\}\!\}_{\mathtt{p}\in\mathcal{P}}$ implements $\mathbf{G}$, both $\bar{w} \cdot \mathtt{q}_2 \triangleright \mathtt{p}!m_2$ and $\bar{w} \cdot \mathtt{q}_2 \triangleright \mathtt{p}!m_2 \cdot \mathtt{p} \triangleleft \mathtt{q}_2?m_2$ are traces of $\{\!\{B_\mathtt{p}\}\!\}_{\mathtt{p}\in\mathcal{P}}$. Let $(\vec{s}'', \xi'')$ and $(\vec{s}''', \xi''')$ respectively denote $\{\!\{B_\mathtt{p}\}\!\}_{\mathtt{p}\in\mathcal{P}}$ configurations such that

$$(\vec{s}_0, \xi_0) \xrightarrow{\bar{w}\cdot\mathtt{q}_2\triangleright\mathtt{p}!m_2} (\vec{s}'', \xi'') \xrightarrow{\mathtt{p}\triangleleft\mathtt{q}_2?m_2\cdot\bar{v}} (\vec{s}''', \xi''') \ .$$

Because send actions are always enabled in a CSM, it suffices to show that $\vec{s}''_{\mathtt{q}_1}$ has an outgoing transition label $\mathtt{q}_1 \triangleright \mathtt{p}!m_1$. We do so by showing that $\vec{s}''_{\mathtt{q}_1} = \vec{s}'''_{\mathtt{q}_1}$: it is clear from the fact that $\bar{w} \cdot \mathtt{q}_2 \triangleright \mathtt{p}!m_2 \cdot \mathtt{p} \triangleleft \mathtt{q}_2?m_2 \cdot \bar{v} \cdot \mathtt{q}_1 \triangleright \mathtt{p}!m_1$ is a trace of $\{\!\{B_\mathtt{p}\}\!\}_{\mathtt{p}\in\mathcal{P}}$ that $\vec{s}'''_{\mathtt{q}_1}$ has an outgoing transition label $\mathtt{q}_1 \triangleright \mathtt{p}!m_1$.

Due to the determinacy of subset construction, it suffices to show that

$$(\bar{w} \cdot \mathtt{q}_2 \triangleright \mathtt{p}!m_2)\Downarrow_{\Sigma_{\mathtt{q}_1}} = (\bar{w} \cdot \mathtt{q}_2 \triangleright \mathtt{p}!m_2 \cdot \mathtt{p} \triangleleft \mathtt{q}_2?m_2 \cdot \bar{v})\Downarrow_{\Sigma_{\mathtt{q}_1}} \ .$$

This equality follows from the definition of M and the fact that $\mathtt{q}_1 \triangleright \mathtt{p}!m_1 \in M^{\mathtt{p}}_{(G_2\ldots)}$: because the blocked set of roles in M monotonically increases, and for any G', B, no actions in a run suffix starting with G' involving roles in B' are included in $M^{B'}_{G'}$, we know that $\mathtt{q}_1 \triangleright \mathtt{p}!m_1$ must be the lexicographically earliest action involving $\mathtt{q}_1$ in $\bar{v} \cdot \mathtt{q}_1 \triangleright \mathtt{p}!m_1 \cdot \mathtt{p} \triangleleft \mathtt{q}_1?m_1$. In other words, $\bar{v}\Downarrow_{\Sigma_{\mathtt{q}_1}} = \varepsilon$.

This concludes the reasoning that $\bar{w}\cdot\mathtt{q}_2\triangleright\mathtt{p}!m_2\cdot\mathtt{q}_1\triangleright\mathtt{p}!m_1$ is a trace of $\{\!\{B_\mathtt{p}\}\!\}_{\mathtt{p}\in\mathcal{P}}$.

Recall that $(\vec{s}', \xi')$ is the $\{\!\{B_\mathtt{p}\}\!\}_{\mathtt{p}\in\mathcal{P}}$ configuration reached on $\bar{w}\cdot\mathtt{q}_2\triangleright\mathtt{p}!m_2\cdot\mathtt{q}_1\triangleright\mathtt{p}!m_1$. We showed above that $\vec{s}_\mathtt{p}$ has two outgoing transitions labeled $\mathtt{p} \triangleleft \mathtt{q}_1?m_1$ and $\mathtt{p}\triangleleft\mathtt{q}_2?m_2$. It follows from the equality below that $\vec{s}'_\mathtt{p}$ likewise has two outgoing transitions labeled $\mathtt{p} \triangleleft \mathtt{q}_1?m_1$ and $\mathtt{p} \triangleleft \mathtt{q}_2?m_2$:

$$(\bar{w} \cdot \mathtt{q}_2 \triangleright \mathtt{p}!m_2)\Downarrow_{\Sigma_\mathtt{p}} = (\bar{w} \cdot \mathtt{q}_2 \triangleright \mathtt{p}!m_2 \cdot \mathtt{q}_1 \triangleright \mathtt{p}!m_1)\Downarrow_{\Sigma_\mathtt{p}} \ .$$

We now show that the channels $\xi'(\mathtt{q}_1, \mathtt{p})$ and $\xi'(\mathtt{q}_2, \mathtt{p})$ respectively contain the messages m_1 and m_2 at the head. Recall that $\bar{w}$ is defined as $\mathtt{split}(\mathtt{trace}(\alpha))$; this from the fact that $\xi_{\bar{w}}$ is uniquely determined by $\bar{w}$ and all channels in $\xi_{\bar{w}}$ are empty.

Let $u := \bar{w}\cdot\mathtt{q}_2\triangleright\mathtt{p}!m_2\cdot\mathtt{q}_1\triangleright\mathtt{p}!m_1$. This concludes our proof that both $u\cdot\mathtt{p}\triangleleft\mathtt{q}_1?m_1$ and $u \cdot \mathtt{p} \triangleleft \mathtt{q}_2?m_2$ are traces of $\{\!\{B_\mathtt{p}\}\!\}_{\mathtt{p}\in\mathcal{P}}$.

End Proof of Claim I.

The next claim establishes that our witness $u \cdot \mathtt{p} \triangleleft \mathtt{q}_1?m_1$ satisfies (b).

Claim II. It holds that $I(u \cdot \mathtt{p} \triangleleft \mathtt{q}_1?m_1) = \emptyset$.

Proof of Claim II. This claim follows trivially from the observation that every run in $I(\bar{w} \cdot \mathtt{q}_2 \triangleright \mathtt{p}!m_2)$ must begin with $\alpha \cdot G_1 \xrightarrow{\mathtt{q}_2\rightarrow\mathtt{p}:m_2} G_2$. Because $I(u \cdot \mathtt{p} \triangleleft \mathtt{q}_1?m_1) \subseteq I(\bar{w} \cdot \mathtt{q}_2 \triangleright \mathtt{p}!m_2)$, and the $\mathtt{split}(\mathtt{trace}(\text{-}))$ of every run in $I(\bar{w} \cdot \mathtt{q}_2 \triangleright \mathtt{p}!m_2)$ starts with $\bar{w} \cdot \mathtt{q}_2 \triangleright \mathtt{p}!m_2 \cdot \mathtt{p} \triangleleft \mathtt{q}_2?m_2$, therefore $I(u \cdot \mathtt{p} \triangleleft \mathtt{q}_1?m_1)$ is empty.

End Proof of Claim II.

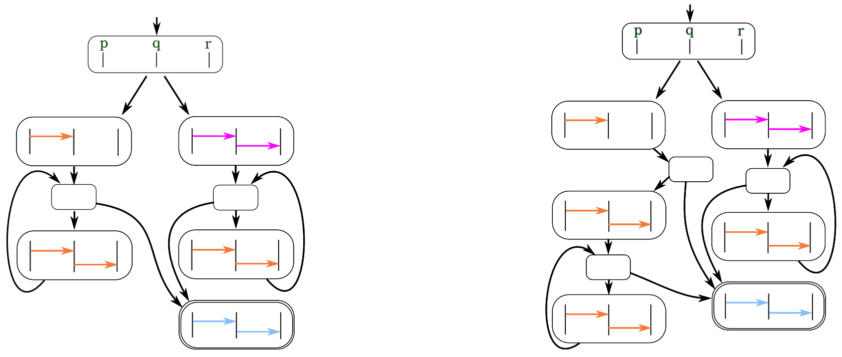
From here, the reasoning that every run in $I(\bar{w} \cdot \mathtt{q}_2 \triangleright \mathtt{p}!m_2)$ must begin with $\alpha \cdot G_1 \xrightarrow{\mathtt{q}_2\rightarrow\mathtt{p}:m_2} G_2$ is identical to the reasoning for the analogous claim in the Send Validity case, and thus omitted.

By choosing $v_0 := \bar{u} \cdot \mathtt{p} \triangleleft \mathtt{q}_1?m_1$, we thus establish both conditions (a) and (b) required for a contradiction. This concludes our proof that Receive Validity is required to hold. $\square$

E Additional Material for §10

E.1 Visual Representations of $\mathbf{G_{fold}}$ and $\mathbf{G_{unf}}$

Figure 4 depicts the examples in §10 visually.



(a) Rejected by syntactic projection operators

(b) Partially unfolded and accepted by syntactic projection operators

Fig. 4: Two protocol specifications for the same protocol

E.2 Entailed Properties from the Literature – Detailed Analysis

For systems with two roles, an *unspecified reception* [14, Def. 12] occurs if a receiver cannot receive the first message in its (only) channel. Intuitively, this yields a deadlock for a binary system or an execution in which the other role will send messages indefinitely. For multi-party systems with sender-driven choice, this is not necessarily the case and our receive validity condition ensures that a message in a role's channel can appear but will not confuse the receiver regarding which choices were made. Thus, we could lift the property to multiparty systems with a universal quantification over channels. Our subset projection would prevent this lifted version of unspecified receptions because of deadlock freedom and the fact that any non-deadlock configuration can be extended in a way that the unspecified reception can eventually be received. Similarly, our subset projection ensures the *absence of orphan messages* [21, Sec. 2] [9, Sec. 3]. However, we need to adapt the definition to our setting. For most MST frameworks, the following two types of CSM configurations are equivalent: final-state configurations, i.e., where each role is in a final state; and sink-state configurations, i.e., where each role is in a state without outgoing transitions. *Orphan messages* have been defined using final-state configurations. Our subset projection is more expressive so both configuration types do not necessarily coincide. Our soundness proof ensures the absence of orphan messages in sink-state configurations and ensures that messages in final-state configurations can be received eventually. We refer to §11 for a discussion on the expressiveness of local types and FSMs.

The standard notion of *progress* [18, Sec. 1] asks that every sent message is eventually received and every process waiting for a message eventually receives one. We proved our subset projection sound for finite as well as infinite CSM runs. For finite runs, both properties trivially hold. For infinite runs, our subset projection ensures that both is possible but one would require fairness assumptions to ensure that it will actually happen as is common for liveness properties.

In our subset projection, it is also guaranteed that each transition of a local implementation can be fired in some execution of the subset projection. This is called *executable* by Cécé and Finkel [14, Def. 12] while it is the property *live* in work by Scalas and Yoshida [36, Fig.5(3)] and called *liveness* by Barbanera et al. [7, Def. 2.9].

A CSM has the *stable property* if any reachable configuration has a transition sequence to a configuration with empty channels. With our proof technique, we showed every run of a CSM has a common path in the protocol that complies with all participants' local observations of the run. There is a furthest point in this path and it is possible that all participants catch up to this point, having empty channels.

Scalas and Yoshida [36] also consider two properties that are rather protocol-specific than implementation-specific, i.e., protocol fidelity and deadlock freedom trivially ensure that every implementation satisfies these properties if the protocol does. First, a global type is *terminating* [36, Fig.5(2)] if every CSM run is finite. It is trivial that this is only true if there are no (used) recursion variable binders μt. Second, a global type is *never-terminating* [36, Fig.5(3)] if every CSM run is infinite. Consequently, this is only the case if the global type has no term 0.