On the Expressiveness of Mixed Choice Sessions

Kirstin Peters
Universität Augsburg
Germany
kirstin.peters@uni-a.de

Nobuko Yoshida
Imperial College London
UK
n.yoshida@imperial.ac.uk

Session types provide a flexible programming style for structuring interaction, and are used to guarantee a safe and consistent composition of distributed processes. Traditional session types include only one-directional input (external) and output (internal) guarded choices. This prevents the session-processes to explore the full expressive power of the π-calculus where the mixed choices are proved more expressive than the (non-mixed) guarded choices. To account this issue, recently Casal, Mordido, and Vasconcelos proposed the binary session types with mixed choices (CMV^+). This paper carries a surprising, unfortunate result on CMV^+: in spite of an inclusion of unrestricted channels with mixed choice, CMV^+'s mixed choice is rather separate and not mixed. We prove this negative result using two methodologies (using either the leader election problem or a synchronisation pattern as distinguishing feature), showing that there exists no good encoding from the π-calculus into CMV^+, preserving distribution. We then close their open problem on the encoding from CMV^+ into CMV (without mixed choice), proving its soundness and thereby that the encoding is good up to coupled similarity.

1 Introduction

Starting with the landmark result by Palamidessi in [21] and followed up by results such as [20, 22, 10, 26, 28, 29] it was shown that the key to the expressive power of the full π-calculus in comparison to its sub-calculi such as e.g. the asynchronous π-calculus is *mixed choice*.

Mixed choice in the π-calculus is a choice construct that allows to choose between inputs and outputs. In contrast, e.g. *separate choices* are constructed from either only inputs or only outputs. The additional expressive power of mixed choice relies on its ability to rule out alternative options of the opposite nature, i.e., a term can rule out its possibility to perform an input by doing an output, whereas without mixed choice inputs can rule out alternative inputs only and outputs may rule out only alternative outputs.

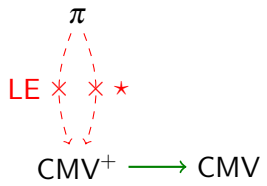
To compare calculi with different variants of choice, we try to build an encoding or show that no such encoding exists [3, 23]. The existence of an encoding that satisfies relevant criteria shows that the target language is expressive enough to emulate the behaviours in the source language. Gorla [10] and others [23, 31] introduced and classified a set of general criteria for encodability which are syntax-agnostic [10, 31]: they are now commonly used for claiming expressiveness of a given calculus, defining important features which a "good encoding" should satisfy. These include *compositionality* (homomorphism), *name invariance* (bijectional renaming), sound and complete *operational correspondence* (the source and target can simulate each other), *divergence reflection* (the target diverges only if the source diverges), *observability* (barb-sensitiveness), and *distributability* preservation (the target has the same degree of distribution as the source). Conversely, a *separation result*, i.e., the proof of the absence of an encoding with certain criteria, shows that the source language can represent behaviours that can*not* be expressed in the target. This paper gives a fresh look at expressiveness of typed π-calculi, focusing on choice constructs of *session types*.

V. Castiglioni and C. A. Mezzina (Eds): Combined Workshop on Expressiveness in Concurrency and Structural Operational Semantics 2022 (EXPRESS/SOS 2022).
EPTCS 368, 2022, pp. 113–130, doi:10.4204/EPTCS.368.7

Session types [13, 37] specify and constrain the communication behaviour as a protocol between components in a system. A session type system excludes any non-conforming behaviour, statically preventing type and communication errors (i.e., mismatch of choice labels). Several languages now have session-type support via libraries and tools [36, 1]. As the origin of session types is Linear Logic [11], traditional session types include only one-directional input (external) and output (internal) guarded choices. To explore the full expressiveness of mixed choice from the π-calculus, recently Casal, Mordido, and Vasconcelos proposed the binary session types with mixed choices called *mixed sessions* [6]. We denote their calculus by CMV^+. Mixed sessions include a mixture of branchings (labelled input choices) and selections (labelled output choices) at the same *linear* channel or *unrestricted* channel. This extension gives us many useful and typable *structured* concurrent programming idioms which consist of both unrestricted and linear non-deterministic choice behaviours. We show that in spite of its practical relevance, mixed sessions in CMV^+ are *strictly less expressive* than mixed choice in the π-calculus even with an unrestricted usage of choice channels.

This result surprised us. We would have expected that using mixed choice with an unrestricted choice channel results into a choice construct comparable to choice in the π-calculus. But, as we show in the following, mixed choice in CMV^+ cannot express essential features of mixed choice in the π-calculus. First we observe that mixed sessions are not expressive enough to solve leader election in symmetric networks. Remember that it was leader election in symmetric networks that was used to show that mixed choice is more expressive than separate choice in the π-calculus (see [21]). Second we observe that mixed sessions cannot express the synchronisation pattern $\star$. Synchronisation patterns were introduced in [31] to capture the amount of synchronisation that can be expressed in distributed systems. The synchronisation pattern $\star$ was identified in [31] as capturing exactly the amount of synchronisation introduced with mixed choice in the π-calculus. Finally, we have a closer look at the encoding from CMV^+ into CMV presented in [6]. CMV is the variant of session types that is extended in [6] with a mixed-choice-construct in order to obtain CMV^+, i.e., CMV has traditional branching and selection but not their mix. As it is the case for many variants of session types, CMV can express separate choice but has no construct for mixed choice. By analysing this encoding, we underpin our claim that mixed choice in CMV^+ is not more expressive than separate choice in the π-calculus.

Our contributions are summarised in the picture on the right. In § 3 we prove that there exists no good encoding from the π-calculus (with mixed choice) into CMV^+, where we use the leader election problem by Palamidessi in [21] (LE) as distinguishing feature (the first - -×- →). In § 4 we reprove this result using the *synchronisation pattern* $\star$ from [31] instead as distinguishing feature (the second - -×- →). Then we prove soundness of the encoding presented in [6] closing their open problem in § 5 (⟶). By this encoding source terms in CMV^+ and their literal translations in CMV are related by *coupled similarity* [25], i.e., CMV^+ is encoded into CMV up to coupled similarity. From the separation results in § 3 and § 4 and the encoding into session types with separate choice in § 5 we conclude that *mixed sessions in [6] can express only separate choice*.



To make our paper readable, we focus on presenting our results with intuitive, self-contained explanations. In particular, we simplify the languages CMV^+ and CMV for the discussion in this paper and omit their type systems. However, the proofs are carried out on the original definitions of the languages from [6]. We include complete proofs of all the statements in this paper and the technical details of the notions from the literature that we use in [34].

2 Technical Preliminaries: Mixed Sessions and Encodability Criteria

This section gives a summary of the π-calculus, CMV^+, CMV, and encodability criteria.

Assume a countably-infinite set $\mathscr{N}$ of *names*. For the π-calculus we additionally assume a set $\{\overline{y} \mid y \in \mathscr{N}\}$ of *co-names*. Let $\tau \notin \mathscr{N} \cup \{\overline{y} \mid y \in \mathscr{N}\}$.

The *syntax* of a process calculus is usually defined by a context-free grammar defining operators. We use $P, Q, \ldots$ to range over process terms. The arguments of a term P that are again process terms are called *subterms* of P. Terms that appear as subterm underneath some (action) prefix are called *guarded*, because the guarded subterm cannot be executed before the guarding action has been performed. Also conditionals, such as if-then-else-constructs, guard their respective subterms. *Expressions* are constructed from variables, unit, and standard boolean operators. We assume an evaluation function $\mathsf{eval}(\cdot)$ that evaluates expressions to *values*.

We assume that the *semantics* is given as an *operational semantics* consisting of inference rules defined on the operators of the language [35]. For many process calculi, the semantics is provided in two forms, as *reduction semantics* and as *labelled transition semantics*. We assume that at least the reduction semantics $\longmapsto$ is given as part of the definition, because its treatment is easier in the context of encodings. A *(reduction) step* is written as $P \longmapsto P'$. If $P \longmapsto P'$, then P' is called *derivative* of P. Let $P \longmapsto$ (or $P \not\longmapsto$) denote the existence (absence) of a step from P, and let $\Longmapsto$ denote the reflexive and transitive closure of $\longmapsto$. A sequence of reduction steps is called a *reduction*. We write $P \longmapsto^{\omega}$ if P has an infinite sequence of steps. We also use *execution* to refer to a reduction starting from a particular term. A process that cannot reduce is called *stuck*.

The application $P\sigma$ of a substitution $\sigma = \{{}^{y_1}/_{x_1}, \ldots, {}^{y_n}/_{x_n}\}$ on a term is defined as the result of simultaneously replacing all free occurrences of x_i by y_i for $i \in \{1, \ldots, n\}$, possibly applying α-conversion to avoid capture or name clashes. For all names in $\mathscr{N} \setminus \{x_1, \ldots, x_n\}$ the substitution behaves as the identity mapping.

2.1 Process and Session Calculi

The π-calculus was introduced by Milner, Parrow, and Walker in [19] and is one of the most well-known process calculi. We consider a variant of the π-calculus with mixed guarded choice and replication but without matching (as in [21]). This variant is often called the synchronous or full π-calculus. *Mixed sessions* are a variant of binary session types introduced by Casal, Mordido, and Vasconcelos in [6] with a choice construct that combines prefixes for sending and receiving. We denote this language as CMV^+. CMV is the session type variant on which CMV^+ is based. A central idea of CMV^+ (and CMV) is that channels are separated in two *channel endpoints* and that interaction is by two processes acting on the respective different ends of such a channel.

The syntax of the π-calculus, CMV^+, and CMV is given as:

$$\mathscr{P}_{\pi}\colon P ::= \sum_{i\in \mathrm{I}} \alpha_i.P_i \mid (\nu x)P \mid P \mid P \mid\ !P \qquad \alpha ::= y(x) \mid \overline{y}z \mid \tau$$

$$\mathscr{P}_{\mathsf{CMV}^+}\colon P ::= y \sum_{i\in \mathrm{I}} M_i \mid P \mid P \mid (\nu yz)P \mid \mathsf{if}\, v\, \mathsf{then}\, P\, \mathsf{else}\, P \mid \mathbf{0} \qquad M ::= \mathrm{l} * v.P \qquad * ::= ! \mid ?$$

$$\mathscr{P}_{\mathsf{CMV}}\colon P ::= y!v.P \mid y?xP \mid x \triangleleft \mathrm{l}.P \mid x \triangleright \{\mathrm{l}_i : P_i\}_{i\in \mathrm{I}} \mid P \mid P \mid (\nu yz)P \mid \mathsf{if}\, v\, \mathsf{then}\, P\, \mathsf{else}\, P \mid \mathbf{0}$$

A choice $\sum_{i\in \mathrm{I}} \alpha_i.P_i$ in $\mathscr{P}_{\pi}$ offers for each i in the index set I a subterm guarded by some action prefix α_i, where α_i is an input action $y(x)$, an output action $\overline{y}z$, or the internal action τ. In contrast choice $y \sum_{i\in \mathrm{I}} M_i$ in CMV^+ operates on a single channel endpoint. In [6] a choice is declared as either linear

(lin) or unrestricted (un), where the latter introduces recursion in the calculus. We omit these qualifiers to simplify the presentation. However, the proofs are carried out on the languages CMV^+ and CMV as given by [6] (see [34]). A branch $\mathrm{l}*v.P$ specifies a label l, a polarity $*$ (! for sending or ? for receiving), a value in output actions or a variable for input actions, and a continuation P. We abbreviate the empty sum by $\mathbf{0}$ and we often separate summands by $+$. The remaining are: restriction $(\nu x)P$ and $(\nu yz)P$, parallel composition $P \mid P$, replication $!P$, conditional $\mathsf{if}\, v\, \mathsf{then}\, P\, \mathsf{else}\, P$, output $y!v.P$, input $y?xP$, selection $x \triangleleft \mathrm{l}.P$, and branching $x \triangleright \{\mathrm{l}_i : P_i\}_{i\in I}$.

The semantics of the languages is given by the axioms:

$$\begin{array}{lc}
\pi\text{:} & \overline{y}z.P + R \mid y(x).Q + N \longmapsto P \mid Q\{^z/_x\} \qquad \tau.P + R \longmapsto P \\
\mathsf{CMV}^+/\mathsf{CMV}\text{:} & \mathsf{if}\,\mathsf{true}\,\mathsf{then}\,P\,\mathsf{else}\,Q \longmapsto P \qquad \mathsf{if}\,\mathsf{false}\,\mathsf{then}\,P\,\mathsf{else}\,Q \longmapsto Q \\
\mathsf{CMV}^+\text{:} & (\nu yz)(y(\mathrm{l}!v.P + M) \mid z(\mathrm{l}?x.Q + N) \mid R) \longmapsto (\nu yz)(P \mid Q\{^v/_x\} \mid R) \\
\mathsf{CMV}\text{:} & (\nu yz)(y!v.P \mid z?xQ \mid R) \longmapsto (\nu yz)(P \mid Q\{^v/_x\} \mid R) \\
 & (\nu yz)\big(y \triangleleft \mathrm{l}_j.P \mid z \triangleright \{\mathrm{l}_i : Q_i\}_{i\in I} \mid R\big) \longmapsto (\nu yz)(P \mid Q_j \mid R)
\end{array}$$

and all three calculi share the following rules:

$$\frac{P \longmapsto P'}{P \mid Q \longmapsto P' \mid Q} \qquad \frac{P \longmapsto P'}{(\nu \hat{x})P \longmapsto (\nu \hat{x})P'} \qquad \frac{P \equiv Q \quad Q \longmapsto Q' \quad Q' \equiv P'}{P \longmapsto P'}$$

where $\hat{x}$ consists of either one or two names; and $\equiv$ denotes the standard *structural congruence* from [6] plus a rule for replication in the π-calculus. More precisely, structural congruence is defined as the least congruence that contains α-conversion and satisfies the rules:

$$(\nu \hat{x})\mathbf{0} \equiv \mathbf{0} \qquad (\nu \hat{x})(\nu \hat{y})P \equiv (\nu \hat{y})(\nu \hat{x})P \qquad P \mid (\nu \hat{x})Q \equiv (\nu \hat{x})(P \mid Q) \quad \text{if } \{\hat{x}\} \cap \mathsf{fn}(P) = \emptyset$$
$$(\nu yz)P \equiv (\nu zy)P \quad P \mid \mathbf{0} \equiv P \quad P \mid Q \equiv Q \mid P \quad P \mid (Q \mid R) \equiv (P \mid Q) \mid R \quad !P \equiv P \mid !P$$

The name x is bound in P by $y(x).P$, $\mathrm{l}?x.P$, and $(\nu x)P$, $(\nu xy)P$, or $(\nu yx)P$. All other names are free. We often omit $\mathbf{0}$ and the argument of action prefixes if it is irrelevant, i.e., we write $y.P$ instead of $y(x).P$ if $x \notin \mathsf{fn}(P)$; and $\overline{y}.P$ instead of $\overline{y}z.P$ if for all matching receivers $y(x).Q$ we have $x \notin \mathsf{fn}(Q)$.

A process P emits a barb $\overline{y}$, denoted as $P{\downarrow_{\overline{y}}}$, if P (in the π-calculus) has an unguarded output $\overline{y}z.P'$ on a free channel $y \in \mathsf{fn}(P)$. Similarly, P has a barb y, denoted as $P{\downarrow_y}$, if P (in the π-calculus) has an unguarded input $y(x).P'$ or if P (in $\mathsf{CMV}^+/\mathsf{CMV}$) has an unguarded choice $y\sum_{i\in I} M_i$, output $y!v.P$, input $y?x.P$, selection $y \triangleleft \mathrm{l}.P$, or branching $y \triangleright \{\mathrm{l}_i : P_i\}_{i\in I}$ on a free $y \in \mathsf{fn}(P)$. In CMV^+ and CMV we do not distinguish between input barbs $\downarrow_y$ and output barbs $\downarrow_{\overline{y}}$ but instead have barbs on different channel end points. The term P reaches a barb β (with $\beta = y$ or $\beta = \overline{y}$), denoted as $P{\Downarrow_\beta}$, if there is some P' such that $P \Longmapsto P'$ and $P'{\downarrow_\beta}$.

The type systems of CMV^+ and CMV in [6] ensure that the two endpoints of a channel are dual, e.g. if one endpoint sends the other has to receive. For the expressiveness results on the choice construct proved in this paper, the type system is not crucial. Indeed, our separation result (in both ways to prove it) is carried out on the untyped version of CMV^+. See [6] or [34] for the full typing systems.

Two terms of a language are usually compared using some kind of a behavioural simulation relation. The most commonly known behavioural simulation relation is bisimulation. A relation $\mathscr{R}$ is a bisimulation if any two related processes mutually simulate their respective sequences of steps, such that the derivatives are again related.

Definition 2.1 (Bisimulation)**.** $\mathscr{R}$ is a (weak reduction, barbed) bisimulation if for each $(P,Q) \in \mathscr{R}$:

- $P \Longmapsto P'$ implies $\exists Q'.\ Q \Longmapsto Q' \wedge (P',Q') \in \mathscr{R}$
- $Q \Longmapsto Q'$ implies $\exists P'.\ P \Longmapsto P' \wedge (P',Q') \in \mathscr{R}$
- $P\Downarrow_\beta$ iff $Q\Downarrow_\beta$ for all barbs β

Two terms are bisimilar if there exists a bisimulation that relates them. For a language $\mathscr{L}$, let $\approx_{\mathscr{L}}$ denote bisimilarity on $\mathscr{L}$.

Another interesting behavioural simulation relation is coupled similarity. It was introduced in [25] and discussed e.g. in [2]. It is strictly weaker than bisimilarity. As pointed out in [25], in contrast to bisimilarity it essentially allows for intermediate states (see § 5). Each symmetric coupled simulation is a bisimulation.

Definition 2.2 (Coupled Simulation)**.** A relation $\mathscr{R}$ is a (weak reduction, barbed) coupled simulation if for each $(P,Q) \in \mathscr{R}$:

- $P \Longmapsto P'$ implies $\exists Q'.\ Q \Longmapsto Q' \wedge (P',Q') \in \mathscr{R}$
- $P \Longmapsto P'$ also implies $\exists Q'.\ Q \Longmapsto Q' \wedge (Q',P') \in \mathscr{R}$
- $P\Downarrow_\beta$ implies $Q\Downarrow_\beta$ for all barbs β

Two terms are coupled similar if they are related by a coupled simulation in both directions.

For all languages considered here, a process P is distributable into $P_1,\ldots,P_n$ if and only if we have $P \equiv (\nu\tilde{y})\,(P_1 \mid \ldots \mid P_n)$ (compare to the notion of a standard form of the π-calculus in [18] and the discussion on distributability in [31]). Moreover, two steps $a : P \longmapsto P_a$ and $b : P \longmapsto P_b$ of P are in conflict if one disables the other, i.e., if a and b compete for some action prefix. More precisely, two steps in the π-calculus are in conflict if they reduce the same choice, two steps in CMV^+ are in conflict if they reduce the same choice or the same conditional, and two steps in CMV are are in conflict if the reduce the same output, input, selection prefix, branching prefix, or conditional. Note that reducing the same choice not necessarily means to reduce the same summand in this choice. The steps a and b are distributable, if P is distributable into at least two parts such that one part performs the step a and the other part performs b.

2.2 Encodability Criteria

An encoding $[\![\cdot]\!]$ is a function from the processes of the source language into the processes of the target language, where we need encodability criteria to rule out trivial or meaningless encodings. In order to provide a general framework, Gorla in [10] suggests five criteria well suited for language comparison. Other frameworks were introduced e.g. in [8, 40]. We replace success sensitiveness of [10] by the stricter barb-sensitiveness, because it is more intuitive. As we claim, all separation results in this paper remain valid w.r.t. success sensitiveness instead of barb-sensitiveness.

The papers [6] and [21] require as additional criterion that the parallel operator is translated homomorphically. To strengthen the separation results, we use the slightly weaker criterion ‘preservation of distributability’ (see [28, 26]). The encoding of [6] that we discuss in § 5 translates the parallel operator homomorphically.

Definition 2.3 (Good Encoding)**.** We consider an encoding $[\![\cdot]\!]$ to be *good* if it is

compositional: The translation of an operator is captured by a context that takes as arguments the translations of the subterms of the operator.

name invariant: For every S and every substitution σ, it holds that $[\![S\sigma]\!] \asymp [\![S]\!]\,\sigma$.

operationally complete: For all $S \Longmapsto_{\mathrm{S}} S'$, it holds $[\![S]\!] \Longmapsto_{\mathrm{T}} \asymp [\![S']\!]$.

operationally sound: For all $[\![S]\!] \Longmapsto_{\mathrm{T}} T$, there is an S' s.t. $S \Longmapsto_{\mathrm{S}} S'$ and $T \Longmapsto_{\mathrm{T}} \asymp [\![S']\!]$.

divergence reflecting: For every S, $[\![S]\!] \longmapsto_{\mathrm{T}}^{\omega}$ implies $S \longmapsto_{\mathrm{S}}^{\omega}$.

barb-sensitive: For every S and every barb y, $S\Downarrow_y$ iff $[\![S]\!]\Downarrow_y$.

distributability preserving: For every $S \in \mathscr{P}_{\mathrm{S}}$ and for all terms $S_1,\ldots,S_n \in \mathscr{P}_{\mathrm{S}}$ that are distributable within S there are some $T_1,\ldots,T_n \in \mathscr{P}_{\mathrm{T}}$ that are distributable within $[\![S]\!]$ such that $T_i \asymp [\![S_i]\!]$ for all $1 \le i \le n$.

Moreover the equivalence $\asymp$ is a barb respecting (weak) reduction bisimulation.

Operational correspondence is the combination of operational *completeness* and *soundness*. Since we are focusing on separation results on untyped languages, we do not require an explicit criterion for types. In [6] the encoding from CMV^+ into CMV is proven to be type sound.

3 Separating Mixed Sessions and the Pi-Calculus via Leader Election

The first expressiveness result on the π-calculus that focuses on mixed choice is the separation result by Palamidessi in [21, 22]. This result uses the problem of leader election in symmetric networks as distinguishing feature.

Following [21] we assume that the set of names $\mathscr{N}$ contains names that identify the processes of the network and that are never used as bound names within electoral systems. For simplicity, we use natural numbers for this kind of names. A leader is announced by unguarding an output on its id. Then a network $P = (\nu\tilde{x})(P_1 \mid \ldots \mid P_k)$ in $\mathscr{P}_\pi$ or $P = (\nu\tilde{x}\tilde{y})(P_1 \mid \ldots \mid P_k)$ in $\mathscr{P}_{\mathsf{CMV}^+}$ is an *electoral system* if in every maximal execution exactly one leader is announced. We adapt the definition of electoral systems of [21] to obtain electoral systems in the π-calculus and in CMV^+.

Definition 3.1 (Electoral System)**.** A network $P = (\nu\tilde{x})(P_1 \mid \ldots \mid P_k)$ in $\mathscr{P}_\pi$ or $P = (\nu\tilde{x}\tilde{y})(P_1 \mid \ldots \mid P_k)$ in $\mathscr{P}_{\mathsf{CMV}^+}$ is an *electoral system* if for every execution $E : P \Longmapsto P'$ there exists an extension $E' : P \Longmapsto P' \Longmapsto P''$ and some $n \in \{1,\ldots,k\}$ (the leader) such that $P'''\downarrow_n$ for all P''' with $P'' \Longmapsto P'''$, but $P'' \not\Downarrow_m$ for any $m \in \{1,\ldots,k\}$ with $m \neq n$.

Accordingly, an electoral system in the π-calculus announces a leader by unguarding some output on n that cannot be reduced or removed, where n is the id of the leader. In CMV^+ a leader is announced by unguarding a choice on the channel n. Since n is free this choice cannot be removed. A network is an electoral system if in every maximal execution exactly one leader n is announced.

We adapt the definition of hypergraphs that are associated to a network of processes in the π-calculus defined in [21] to networks in CMV^+. The hypergraph connects the nodes $1,\ldots,k$ of the network by edges representing the free channels that they share, where we ignore the outer restrictions of the network.

Definition 3.2 (Hypergraph)**.** Given a network $P = (\nu\tilde{x})(P_1 \mid \ldots \mid P_k)$ in $\mathscr{P}_\pi$ or $P = (\nu\tilde{x}\tilde{y})(P_1 \mid \ldots \mid P_k)$ in $\mathscr{P}_{\mathsf{CMV}^+}$, the *hypergraph* associated to P is $\mathsf{H}(P) = \langle N, X, t\rangle$ with $N = \{1,\ldots,k\}$, $X = \mathsf{fn}(P_1 \mid \ldots \mid P_n) \setminus N$, and $t(x) = \{n \mid x \in \mathsf{fn}(P_n)\}$ for each $x \in X$.

Because we ignore the outer restrictions of the network in the above definition, the hypergraphs of two structural congruent networks may be different. However, this is not crucial for our results.

Given a hypergraph $H = \langle N, X, t \rangle$, an automorphism on H is a pair $\sigma = \langle \sigma_N, \sigma_X \rangle$ such that $\sigma_N : N \to N$ and $\sigma_X : X \to X$ are permutations which preserve the type of arcs. For simplicity, we usually do not distinguish between σ_N and σ_X and simply write σ. Moreover, since σ is a substitution, we allow to apply σ on terms P, denoted as $P\sigma$. The orbit $O_\sigma(n)$ of $n \in N$ generated by σ is defined as the set of nodes in which the various iterations of σ map n, i.e., $O_\sigma(n) = \{n, \sigma(n), \ldots, \sigma^{h-1}(n)\}$, where σ^i represents the composition of σ with itself i times and $\sigma^h = \mathsf{id}$. We also adapt the notion of a symmetric system of [21] to obtain symmetric systems in the π-calculus as well as in CMV^+.

Definition 3.3 (Symmetric System)**.** Consider a network $P = (\nu \tilde{x})(P_1 \mid \ldots \mid P_k)$ in $\mathscr{P}_\pi$ or a network $P = (\nu \tilde{x}\tilde{y})(P_1 \mid \ldots \mid P_k)$ in $\mathscr{P}_{\mathsf{CMV}^+}$, and let σ be an isomorphism on its associated hypergraph $\mathsf{H}(P) = \langle N, X, t \rangle$. P is *symmetric w.r.t.* σ iff $P_{\sigma(i)} \approx_\pi P_i\sigma$ or $P_{\sigma(i)} \approx_{\mathsf{CMV}^+} P_i\sigma$ for each node $i \in N$. P is *symmetric* if it is symmetric w.r.t. all the automorphisms of $\mathsf{H}(P)$.

In contrast to [21] we use bisimilarity—$\approx_\pi$ and $\approx_{\mathsf{CMV}^+}$—instead of alpha conversion in the definition of symmetry. With this weaker notion of symmetry, we compensate for the weaker criterion on distributability that we use instead of the homomorphic translation of the parallel operator. Accordingly, we also consider networks as symmetric if they behave in a symmetric way; they do not necessarily need to be structurally symmetric.

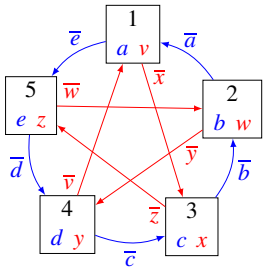
In the π-calculus we find symmetric electoral systems for many kinds of hypergraphs. We use such a solution of leader election in a network with five nodes as counterexample to separate CMV^+ from the π-calculus.

Example 3.4 (Leader Election in the π-Calculus)**.** Consider the network

$$\mathsf{S}_\pi^{\mathsf{LE}} = (\nu a,b,c,d,e,v,w,x,y,z)\,(S_1 \mid S_2 \mid S_3 \mid S_4 \mid S_5)$$

where $S_1 = \overline{e} + a.\big(\overline{x} + v.\overline{1}\big)$, $S_2 = \overline{a} + b.\big(\overline{y} + w.\overline{2}\big)$, $S_3 = \overline{b} + c.\big(\overline{z} + x.\overline{3}\big)$, $S_4 = \overline{c} + d.\big(\overline{v} + y.\overline{4}\big)$, and $S_5 = \overline{d} + e.\big(\overline{w} + z.\overline{5}\big)$. □

$\mathsf{S}_\pi^{\mathsf{LE}}$ is symmetric. Consider e.g. the permutation σ that permutes the channels as follows: $a \to b \to c \to d \to e \to a$, $v \to w \to x \to y \to z \to v$, and $1 \to 2 \to 3 \to 4 \to 5 \to 1$. Then $S_{\sigma(i)} = S_i\sigma$ for all $i \in \{1, \ldots, 5\}$. The network elects a leader in two stages. The first stage (depicted as blue circle) uses mixed choices on the channels a, b, c, d, e; in the second stage (depicted as a red star) we have mixed choices on the channels v, w, x, y, z. The picture on the right gives $\mathsf{H}\big(\mathsf{S}_\pi^{\mathsf{LE}}\big)$ extended by arrow heads to visualise the direction of interactions and the respective action prefixes. The senders in the two stages are losing the leader election game, i.e., are not becoming the leader. In the first stage two processes can be receivers and continue with the second stage. The process that is neither sender nor receiver in the first stage is stuck and also loses. The receiver of the second stage then becomes the leader by unguarding an output on its id. For instance we obtain the execution



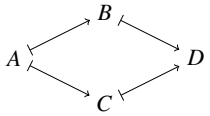
$$\mathsf{S}_\pi^{\mathsf{LE}} \longmapsto (\nu \tilde{n})\big(\overline{x} + v.\overline{1} \mid S_3 \mid S_4 \mid S_5\big) \longmapsto (\nu \tilde{n})\big(\overline{x} + v.\overline{1} \mid \overline{z} + x.\overline{3} \mid S_5\big) \longmapsto \overline{3} \mid (\nu \tilde{n})S_5 \not\longmapsto$$

with $\tilde{n} = a,b,c,d,e,v,w,x,y,z$ by reducing on the channels a and c in the first stage. The network $\mathsf{S}_\pi^{\mathsf{LE}}$ has 10 maximal executions (modulo structural congruence) that are obtained from the above execution by symmetry on the first two steps. In each maximal execution exactly one leader is elected.

We show that there exists no symmetric electoral system for networks of size five in CMV^+; or more generally no symmetric electoral system for networks of odd size in CMV^+. A key ingredient to separate the π-calculus with mixed choice from the asynchronous π-calculus in [21] is a confluence lemma. It states that in the asynchronous π-calculus a step reducing an output and an alternative step reducing an input cannot be conflict to each other and thus can be executed in any order. In the full π-calculus this confluence lemma is not valid, because inputs and outputs can be combined within a single choice construct and can thus be in conflict. For CMV^+ we observe that steps that reduce different endpoints can also not be in conflict to each other, because different channel endpoints cannot be combined in a single choice.

Lemma 3.5 (Confluence)**.** *Let $P,Q \in \mathscr{P}_{\mathsf{CMV}^+}$. Assume that $A = (\nu\tilde{x}\tilde{y})(P \mid Q)$ can make two steps $A \longmapsto (\nu\widetilde{x_1}\widetilde{y_1})(P_1 \mid Q_1) = B$ and $A \longmapsto (\nu\widetilde{x_2}\widetilde{y_2})(P_2 \mid Q_2) = C$ such that P_1 is obtained modulo $\equiv$ from P by reducing a choice on channel endpoint a and P_2 is obtained modulo $\equiv$ from P by reducing a choice on channel endpoint b with $a \neq b$. Then there exist $P_3, Q_3 \in \mathscr{P}_{\mathsf{CMV}^+}$ and $D = (\nu\widetilde{x_3}\widetilde{y_3})(P_3 \mid Q_3)$ such that $B \longmapsto D$ and $C \longmapsto D$, where $\widetilde{x_3} = \widetilde{x_1} \cup \widetilde{x_2}$ and $\widetilde{y_3} = \widetilde{y_1} \cup \widetilde{y_2}$.*

The proof of this confluence lemma relies on the observation that the two steps of A to B and C have to reduce distributable parts of A. Then these two steps are distributable, which in turn allows us to perform them in any order. Thus the expressive power of choice in CMV^+ is limited by the fact that syntactically the choice construct is fixed on a single channel endpoint. With this alternative confluence lemma, we can show that there is no electoral system of odd degree in CMV^+.



Lemma 3.6 (No Electoral System)**.** *Consider a network $P = (\nu\tilde{x}\tilde{y})(P_1 \mid \ldots \mid P_k)$ in CMV^+ with $k > 1$ being an odd number. Assume that the associated hypergraph $\mathsf{H}(P)$ admits an automorphism $\sigma \neq \mathsf{id}$ with only one orbit, and that P is symmetric w.r.t. σ. Then P cannot be an electoral system.*

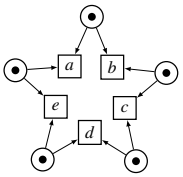
In the proof we construct a potentially infinite sequence of steps such that the system constantly restores symmetry, i.e., whenever a step destroys symmetry we can perform a sequence of steps that restores the symmetry. Therefore we rely on the assumption of σ generating only one orbit. This implies that $O_\sigma(i) = \{i, \sigma(i), \ldots, \sigma^{k-1}(i)\} = \{1, \ldots, k\}$, for each $i \in \{1, \ldots, k\}$. Because of that, whenever part i performs a step that destroys symmetry or parts i and j together perform a step that destroys symmetry, the respective other parts of the originally symmetric network can perform symmetric steps to restore the symmetry of the network. Because of the symmetry, the constructed sequence of steps does not elect a unique leader. Accordingly, the existence of this sequence ensures that P is not an electoral system. The odd degree of the network is necessary to ensure that we can apply the confluence lemma, which in turn ensures that we can always perform the sequence of steps to restore symmetry after the step that destroys the symmetry.

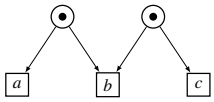
By the preservation of distributability, encodings preserve the structure of networks; and by name invariance, they also preserve the symmetry of networks. With operational correspondence and barb-sensitiveness, any good encoding of $\mathsf{S}_\pi^{\mathsf{LE}}$ is again a symmetric electoral system of size five. Since by Lemma 3.6 this is not possible, we can separate CMV^+ from the π-calculus.

Theorem 3.7 (Separate CMV^+ from the π-Calculus via Leader Election)**.**
There is no good encoding from the π-calculus into CMV^+.

4 Separating Mixed Sessions and the Pi-Calculus via Synchronisation

In [31] the technique used in [21] and its relation to synchronisation are analysed. Two synchronisation patterns, the pattern M and the pattern $\star$, are identified that describe two different levels of synchronisation and allow to more clearly separate languages along their ability to express synchronisation. These patterns are called M and $\star$, because their respective representations as a Petri net (see left and right picture) have these shapes. The pattern $\star$ captures the power of synchronisation of the π-calculus. In particular it captures what is necessary to solve the leader election problem.





The pattern M captures a very weak form of synchronisation, not enough to solve leader election but enough to make a fully distributed implementation of languages with this pattern difficult (see also [30]). This pattern was originally identified in [38] when studying the relevance of synchrony and distribution on Petri nets. As shown in [26, 31], the ability to express these different amounts of synchronisation in the π-calculus lies in its different forms of choices: to express the pattern $\star$ the π-calculus needs mixed choice, whereas separate choice allows to express the pattern M. Indeed we find the pattern M in CMV^+, but there are no $\star$ in CMV^+.

Example 4.1 (A M in CMV^+)**.** The process $\mathsf{P}_{\mathsf{M}}^{\mathsf{CMV}^+}$ is a M in CMV^+:

$$\mathsf{P}_{\mathsf{M}}^{\mathsf{CMV}^+} = (\nu xy)\Big(\underbrace{\begin{array}{l} x(1!\mathsf{true}.P_1 + 1?z.P_2) \\ y(1?z.P_5 + 1!\mathsf{true}.P_6)\end{array}}_{\text{location 1}} \mid \underbrace{\begin{array}{l} x(1!\mathsf{false}.P_3 + 1?z.P_4) \\ y(1?z.P_7 + 1!\mathsf{false}.P_8)\end{array}}_{\text{location 2}} \mid \Big)$$

A process is a M if it can perform three steps a,b,c, where a,b,c are names and not labels, such that a and b as well as b and c are in conflict whereas a and c are distributable steps. For instance we can pick the steps a, b, and c as:

Step a**:** $\mathsf{P}_{\mathsf{M}}^{\mathsf{CMV}^+} \longmapsto (\nu xy)\,(P_1 \mid x(1!\mathsf{false}.P_3 + 1?z.P_4) \mid P_5\{^{\mathsf{true}}/_z\} \mid y(1?z.P_7 + 1!\mathsf{false}.P_8))$

Step b**:** $\mathsf{P}_{\mathsf{M}}^{\mathsf{CMV}^+} \longmapsto (\nu xy)\,(P_1 \mid x(1!\mathsf{false}.P_3 + 1?z.P_4) \mid y(1?z.P_5 + 1!\mathsf{true}.P_6) \mid P_7\{^{\mathsf{true}}/_z\})$

Step c**:** $\mathsf{P}_{\mathsf{M}}^{\mathsf{CMV}^+} \longmapsto (\nu xy)\,(x(1!\mathsf{true}.P_1 + 1?z.P_2) \mid P_3 \mid y(1?z.P_5 + 1!\mathsf{true}.P_6) \mid P_7\{^{\mathsf{false}}/_z\})$

The process $\mathsf{P}_{\mathsf{M}}^{\mathsf{CMV}^+}$ is well-typed (see [34]). □

We use synchronisation patterns and the proof technique presented in [31] to present an alternative way to prove Theorem 3.7. By that we underpin our claim that the choice construct of CMV^+ is separate and not mixed, and we provide further intuition on why this choice construct is less expressive.

We inherit the definition of the synchronisation pattern $\star$ from [31], where we do not distinguish between local and non-local $\star$ since in the π-calculus there is no difference between parallel and distributable steps.

Definition 4.2 (Synchronisation Pattern $\star$)**.** Let $\langle \mathscr{P}, \longmapsto \rangle$ be a process calculus and $\mathsf{P}^\star \in \mathscr{P}$ such that:

- $\mathsf{P}^\star$ can perform at least five alternative reduction steps $i : \mathsf{P}^\star \longmapsto P_i$ for $i \in \{a,b,c,d,e\}$ such that the P_i are pairwise different;
- the steps a, b, c, d, and e form a circle such that a is in conflict with b, b is in conflict with c, c is in conflict with d, d is in conflict with e, and e is in conflict with a; and

- every pair of steps in $\{a,b,c,d,e\}$ that is not in conflict due to the previous condition is distributable in $\mathsf{P}^\star$.

In this case, we denote the process $\mathsf{P}^\star$ as $\star$.

In contrast to CMV^+ we do find $\star$ in the π-calculus.

Example 4.3 (The $\star$ in the π-Calculus)**.** Consider the following $\star$ in the π-calculus:

$$\mathsf{S}^\star_\pi = \overline{a} + b.\overline{o_b} \mid \overline{b} + c.\overline{o_c} \mid \overline{c} + d.\overline{o_d} \mid \overline{d} + e.\overline{o_e} \mid \overline{e} + a.\overline{o_a}$$

The steps $a,\ldots,e$ of Definition 4.2 are the steps on the respective channels.

Step a**:** $\mathsf{S}^\star_\pi \longmapsto S_a$ with $S_a = \overline{b} + c().\overline{o_c} \mid \overline{c} + d().\overline{o_d} \mid \overline{d} + e().\overline{o_e} \mid \overline{o_a}$,

Step b**:** $\mathsf{S}^\star_\pi \longmapsto S_b$ with $S_b = \overline{o_b} \mid \overline{c} + d().\overline{o_d} \mid \overline{d} + e().\overline{o_e} \mid \overline{e} + a().\overline{o_a}$,

Step c**:** $\mathsf{S}^\star_\pi \longmapsto S_c$ with $S_c = \overline{a} + b().\overline{o_b} \mid \overline{o_c} \mid \overline{d} + e().\overline{o_e} \mid \overline{e} + a().\overline{o_a}$,

Step d**:** $\mathsf{S}^\star_\pi \longmapsto S_d$ with $S_d = \overline{a} + b().\overline{o_b} \mid \overline{b} + c().\overline{o_c} \mid \overline{o_d} \mid \overline{e} + a().\overline{o_a}$

Step e**:** $\mathsf{S}^\star_\pi \longmapsto S_e$ with $S_e = \overline{a} + b().\overline{o_b} \mid \overline{b} + c().\overline{o_c} \mid \overline{c} + d().\overline{o_d} \mid \overline{o_e}$

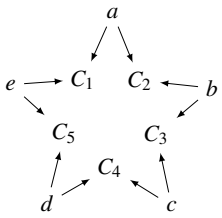
The different outputs $\overline{o_x}$ allow to distinguish between the different steps by their observables. □

We use the $\star$ $\mathsf{S}^\star_\pi$ as counterexample to show that there is no good encoding from the π-calculus into CMV^+. From Lemma 3.6 we learned that CMV^+ cannot express certain electoral systems. Accordingly, we are not surprised that CMV^+ cannot express the pattern $\star$.

Lemma 4.4. *There are no* $\star$ *in* CMV^+.

Proof. Assume the contrary, i.e., assume that there is a term $\mathsf{P}^\star_{\mathsf{CMV}^+}$ in CMV^+ that is a $\star$. Then $\mathsf{P}^\star_{\mathsf{CMV}^+}$ can perform at least five alternative reduction steps a,b,c,d,e such that neighbouring steps in the sequence a,b,c,d,e,a are pairwise in conflict and non-neighbouring steps are distributable. Since steps reducing a conditional cannot be in conflict with any other step, none of the steps in $\{a,b,c,d,e\}$ reduces a conditional. Then all steps in $\{a,b,c,d,e\}$ are communication steps that reduce an output and an input that both are part of choices (with at least one summand). Because of the conflict between a and b, these two steps reduce the same choice but this choice is not reduced in c, because a and c are distributable.

By repeating this argument, we conclude that in the steps a,b,c,d,e five choices $C_1,\ldots,C_5$ are reduced as depicted on the right, where e.g. the step a reduces the choices C_1 and C_2. By the reduction semantics of CMV^+, the two choices C_1 and C_2 that are reduced in step a need to use dual endpoints of the same channel. Without loss of generality, assume that C_1 is on channel endpoint x and C_2 is on channel endpoint y. Then the choice C_3 needs to be on channel endpoint x again, because step b reduces C_2 (on y) and C_3. By repeating this argument, then C_4 is on y and C_5 is on x. But then step e reduces two choices C_1 and C_5 that are both on channel endpoint x. Since the reduction semantics of CMV^+ does not allow such a step, this is a contradiction.



We conclude that there are no $\star$ in CMV^+. □

The proof of the above lemma tells us more about why choice in CMV^+ is limited. From the confluence property in CMV^+ we get the hint that the problem is the restriction of choice to a single channel endpoint. A $\star$ is a circle of steps of odd degree, where neighbouring steps are in conflict. More precisely, the star with five points in $\star$ is the smallest cycle of steps where neighbouring steps are in conflict and that

contains non-neighbouring distributable steps. The proof shows that the limitation of choice to a single channel endpoint and the requirement of the semantics that a channel endpoint can interact with exactly one other channel endpoint causes the problem. This also explains why Lemma 3.6 considers electoral systems of odd degree, because the odd degree does not allow to close the cycle as explained in the proof above. Indeed, if we change the syntax to allow mixed choice with summands on more than one channel, we obtain the mixed-choice-construct of the π-calculus. Similarly, we invalidate our separation result in the Theorems 3.7 and 4.5, if we change the semantics to allow two choices to communicate even if they are on the same channel. The latter may be more surprising, but indeed we do not need more than a single channel to solve leader election and build $\star$, e.g. $\mathsf{S}^{\star}_{\pi}$ remains a star if we choose $a = b = c = d = e$ (though we might want to pick different names $o_a, \ldots, o_e$ to be able to distinguish the steps).

We use $\mathsf{S}^{\star}_{\pi}$ in Example 4.3 as counterexample.

Theorem 4.5 (Separate CMV^{+} and the π-Calculus via $\star$)**.**
There is no good and distributability preserving encoding from the π-calculus into CMV^{+}.

To prove the above theorem, we show that the conflicts in the counterexample $\mathsf{S}^{\star}_{\pi}$ have to be translated into conflicts in its literal translation. Since the target language CMV^{+} cannot express a $\star$, to emulate $\mathsf{S}^{\star}_{\pi}$ it has to break the cycle and split at least one of the conflicts with the respective two neighbouring steps into two distributable conflicts: one for each neighbour. This causes a contradiction, because the distribution of the conflict induces new behaviour that is observable modulo the criteria we picked for good encodings.

5 Encoding Mixed Sessions into Separate Choice

In [6, § 7] an encoding of mixed sessions (CMV^{+}) into the variant of this session type system CMV with only separate choice (branching and selection) is presented. The proof of soundness of this encoding is missing in [6]. They suggest to prove soundness modulo "a weak form of bisimulation". As discussed below, the soundness criterion used in [6] needs to be corrected first.

The main idea of $[\![\cdot]\!]^{\mathsf{CMV}^{+}}_{\mathsf{CMV}}$ is to encode the information about whether a summand is an output or an input into the label used in branching, where a label l_i used with polarity ! in a choice typed as internal becomes $\mathrm{l}_{i,!}$ and in a choice typed as external it becomes $\mathrm{l}_{i,?}$. The dual treatment of polarities w.r.t. the type ensures that the labels of matching communication partners are translated to the same label.

Example 5.1 (Translation)**.** Consider for example the term $S \in \mathscr{P}_{\mathsf{CMV}^{+}}$:

$$S = (\nu xy)\,(y\,(\mathrm{l}!\mathsf{false}.S_1 + \mathrm{l}?z.S_2) \mid x\,(\mathrm{l}!\mathsf{true}.\mathbf{0} + \mathrm{l}?z.\mathbf{0}) \mid y\,(\mathrm{l}!\mathsf{false}.S_3 + \mathrm{l}?z.S_4))$$

S is well-typed but the type system forces us to assign dual types to x and y. Because of that, the choices on one channel need to be internal and on the other external. Let us assume that we have external choices on y and that the choice on x is internal. Moreover, we assume that both channels are marked as linear but typed as unrestricted. Then the translation[1] yields $[\![S]\!]^{\mathsf{CMV}^{+}}_{\mathsf{CMV}} \Longmapsto T_1$ with

$$\begin{aligned} T_1 = (\nu xy)\big(\, & y?c.c \triangleright \Big\{ \mathrm{l}_? : \Big(c!\mathsf{false}.[\![S_1]\!]^{\mathsf{CMV}^{+}}_{\mathsf{CMV}} \mid J_1\Big),\quad \mathrm{l}_! : \Big(c?z.[\![S_2]\!]^{\mathsf{CMV}^{+}}_{\mathsf{CMV}} \mid J_2\Big) \Big\} \\ & \mid (\nu st)\big(\, s \triangleright \{\mathrm{l}_1 : (\nu cd)\,(x!c.d \triangleleft \mathrm{l}_!.(d!\mathsf{true}.\mathbf{0} \mid J_3)),\quad \mathrm{l}_2 : (\nu cd)\,(x!c.d \triangleleft \mathrm{l}_?.(d?z.\mathbf{0} \mid J_4))\} \\ & \qquad \mid t \triangleleft \mathrm{l}_1.\mathbf{0} \mid t \triangleleft \mathrm{l}_2.\mathbf{0}\big) \\ & \mid y?c.c \triangleright \Big\{ \mathrm{l}_? : \Big(c!\mathsf{false}.[\![S_3]\!]^{\mathsf{CMV}^{+}}_{\mathsf{CMV}} \mid J_5\Big),\quad \mathrm{l}_! : \Big(c?z.[\![S_4]\!]^{\mathsf{CMV}^{+}}_{\mathsf{CMV}} \mid J_6\Big) \Big\}\big) \end{aligned}$$

[1] Note that [6] introduces a typed encoding, thus $[\![P]\!]^{\mathsf{CMV}^{+}}_{\mathsf{CMV}}$ actually means $[\![\Gamma \vdash P]\!]^{\mathsf{CMV}^{+}}_{\mathsf{CMV}}$, where $\Gamma \vdash P$ is the type statement ensuring that P is well-typed.

where we already performed a few steps to hide some technical details of the encoding function $[\![\cdot]\!]^{\mathsf{CMV}^+}_{\mathsf{CMV}}$ that are not relevant for this explanation and where the $J_1,\ldots,J_6$ remain as junk from performing these steps. We call terms junk if they are stuck and do not emit barbs, i.e., we can ignore the junk. In particular, junk is invisible modulo $\approx_{\mathsf{CMV}}$. We observe, that in the translation of the first $y\,(\mathsf{l}!\mathsf{false}.S_1 + \mathsf{l}?z.S_2)$ in the first line of T_1 the output with label l is translated to the label $\mathsf{l}_?$ and the input with label l is translated to the label $\mathsf{l}_!$, whereas in the translation of its dual $x\,(\mathsf{l}!\mathsf{true}.\mathbf{0} + \mathsf{l}?z.\mathbf{0})$ in the second line of T_1 we obtain $\mathsf{l}_!$ for the output and $\mathsf{l}_?$ for the input. To emulate the step $S \longmapsto S_2' = (\nu xy)\,(S_2\{^{\mathsf{true}}/_z\} \mid y\,(\mathsf{l}!\mathsf{false}.S_3 + \mathsf{l}?z.S_4))$ of S in that true is transmitted to S_2, we start by picking the corresponding alternative, namely l_1 for sending, in the second and third line of T_1

$$\begin{aligned} T_1 \longmapsto T_2 = (\nu xy)\big(\; & y?c.c \triangleright \Big\{ \mathsf{l}_? : \Big(c!\mathsf{false}.[\![S_1]\!]^{\mathsf{CMV}^+}_{\mathsf{CMV}} \mid J_1\Big),\quad \mathsf{l}_! : \Big(c?z.[\![S_2]\!]^{\mathsf{CMV}^+}_{\mathsf{CMV}} \mid J_2\Big)\Big\} \\ & \mid (\nu cd)\,(x!c.d \triangleleft \mathsf{l}_!.\,(d!\mathsf{true}.\mathbf{0} \mid J_3)) \mid J_7 \\ & \mid y?c.c \triangleright \Big\{ \mathsf{l}_? : \Big(c!\mathsf{false}.[\![S_3]\!]^{\mathsf{CMV}^+}_{\mathsf{CMV}} \mid J_5\Big),\quad \mathsf{l}_! : \Big(c?z.[\![S_4]\!]^{\mathsf{CMV}^+}_{\mathsf{CMV}} \mid J_6\Big)\Big\}\big) \end{aligned}$$

where J_7 again remains as junk. Then we perform a communication on xy, where we chose the input on y in the first line:

$$\begin{aligned} T_2 \longmapsto T_3 = (\nu xy)\big(\; & (\nu cd)\big(\; c \triangleright \Big\{ \mathsf{l}_? : \Big(c!\mathsf{false}.[\![S_1]\!]^{\mathsf{CMV}^+}_{\mathsf{CMV}} \mid J_1\Big),\quad \mathsf{l}_! : \Big(c?z.[\![S_2]\!]^{\mathsf{CMV}^+}_{\mathsf{CMV}} \mid J_2\Big)\Big\} \\ & \mid d \triangleleft \mathsf{l}_!.\,(d!\mathsf{true}.\mathbf{0} \mid J_3)\big) \mid J_7 \\ & \mid y?c.c \triangleright \Big\{ \mathsf{l}_? : \Big(c!\mathsf{false}.[\![S_3]\!]^{\mathsf{CMV}^+}_{\mathsf{CMV}} \mid J_5\Big),\quad \mathsf{l}_! : \Big(c?z.[\![S_4]\!]^{\mathsf{CMV}^+}_{\mathsf{CMV}} \mid J_6\Big)\Big\}\big) \end{aligned}$$

Finally, two more steps on cd resolve the branching and transmit true:

$$\begin{aligned} T_3 \longmapsto\!\!\!\longmapsto T_4 = (\nu xy)\big(\; & [\![S_2]\!]^{\mathsf{CMV}^+}_{\mathsf{CMV}}\{^{\mathsf{true}}/_z\} \mid J_2 \mid J_3 \mid J_7 \mid J_8 \\ & \mid y?c.c \triangleright \Big\{ \mathsf{l}_? : \Big(c!\mathsf{false}.[\![S_3]\!]^{\mathsf{CMV}^+}_{\mathsf{CMV}} \mid J_5\Big),\quad \mathsf{l}_! : \Big(c?z.[\![S_4]\!]^{\mathsf{CMV}^+}_{\mathsf{CMV}} \mid J_6\Big)\Big\}\big) \end{aligned}$$

This completes the emulation of $S \longmapsto S_2'$, i.e., the emulation of the single source term step $S \longmapsto S_2'$ required a sequence of target term steps $[\![S]\!]^{\mathsf{CMV}^+}_{\mathsf{CMV}} \Longmapsto T_1 \longmapsto T_2 \longmapsto T_3 \longmapsto\!\!\!\longmapsto T_4$. □

The operational soundness is defined in [6] as (adapting the notation):

$$\text{If } [\![S]\!] \longmapsto_{\mathrm{T}} T \text{ then } S \longmapsto_{\mathrm{S}} S' \text{ and } T \Longmapsto_{\mathrm{T}} \asymp [\![S']\!]. \tag{1}$$

As visualised above, the encoding translates a single source term step into a sequence of target term steps. Unfortunately, for such encodings the statement in (1) is not strong enough: with (1), we check only that the first step on a literal translation does not introduce new behaviour. The requirement $T \Longmapsto_{\mathrm{T}} \asymp [\![S']\!]$ additionally checks that the emulation started with $[\![S]\!] \longmapsto_{\mathrm{T}} T$ can be completed, but not that there are no alternative steps introducing new behaviour. Hence we prove a correct version of soundness as defined in [10] (see Definition 2.3).

Lemma 5.2 (Soundness, $[\![\cdot]\!]^{\mathsf{CMV}^+}_{\mathsf{CMV}}$)**.** *The encoding $[\![\cdot]\!]^{\mathsf{CMV}^+}_{\mathsf{CMV}}$ is operationally sound modulo $\approx_{\mathsf{CMV}}$, i.e., $[\![S]\!]^{\mathsf{CMV}^+}_{\mathsf{CMV}} \Longmapsto T$ implies $S \Longmapsto S'$ and $T \Longmapsto \approx_{\mathsf{CMV}} [\![S']\!]^{\mathsf{CMV}^+}_{\mathsf{CMV}}$.*

As suggested we use $\approx_{\mathsf{CMV}}$, i.e., a form of weak reduction barbed bisimilarity that we simply call bisimilarity in the following. For soundness we have to show that all steps of encoded terms belong modulo bisimilarity to the emulation of a source term step. To prove Lemma 5.2, we analyse the sequence of steps $[\![S]\!]^{\mathsf{CMV}^+}_{\mathsf{CMV}} \Longmapsto T$ and identify all source term steps $S \Longmapsto S'$ whose emulation is started within

$[\![S]\!]^{\mathsf{CMV}^+}_{\mathsf{CMV}} \Longmapsto T$ and the target term steps $T \Longmapsto\approx_{\mathsf{CMV}} [\![S']\!]^{\mathsf{CMV}^+}_{\mathsf{CMV}}$ that are necessary to complete all started emulations modulo bisimulation. Therefore, we use an induction on the number of steps in the sequence $[\![S]\!]^{\mathsf{CMV}^+}_{\mathsf{CMV}} \Longmapsto T$ and analyse the encoding function in order to distinguish between different kinds of target term steps and the emulations of source term steps to that they belong. Note that, as it is typical for many encodability results, the proof of operational soundness is more elaborate than the proof of operational completeness presented in [6].

In Example 5.1 we have $T_4 \approx_{\mathsf{CMV}} [\![S'_2]\!]^{\mathsf{CMV}^+}_{\mathsf{CMV}}$, because all differences between T_4 and $[\![S'_2]\!]^{\mathsf{CMV}^+}_{\mathsf{CMV}}$ are due to junk that cannot be observed modulo $\approx_{\mathsf{CMV}}$. In fact, we have already $T_3 \approx_{\mathsf{CMV}} [\![S'_2]\!]^{\mathsf{CMV}^+}_{\mathsf{CMV}}$, since we consider a weak form of bisimulation here.

In the above variant of soundness T can catch up with the source term S' by the steps $T \Longmapsto\approx_{\mathsf{CMV}} [\![S']\!]^{\mathsf{CMV}^+}_{\mathsf{CMV}}$. This allows for so-called *intermediate states*: target terms that are strictly in between the translation of two source terms, i.e., T such that $S \longmapsto S'$, $[\![S]\!]^{\mathsf{CMV}^+}_{\mathsf{CMV}} \Longmapsto T \Longmapsto\approx_{\mathsf{CMV}} [\![S']\!]^{\mathsf{CMV}^+}_{\mathsf{CMV}}$, but neither $[\![S]\!]^{\mathsf{CMV}^+}_{\mathsf{CMV}} \approx_{\mathsf{CMV}} T$ nor $[\![S']\!]^{\mathsf{CMV}^+}_{\mathsf{CMV}} \approx_{\mathsf{CMV}} T$ (see [25, 31]). In $[\![\cdot]\!]^{\mathsf{CMV}^+}_{\mathsf{CMV}}$ such intermediate states are caused by mapping the task of finding matching communication partners of a single source term step onto several steps in the target. Consider the term T_2 in the above emulation of $S \longmapsto S'_2$. By picking the branch with label l_1, we discarded the branch with label l_2. Because of that, the emulation starting with $[\![S]\!]^{\mathsf{CMV}^+}_{\mathsf{CMV}} \Longmapsto T_2$ can no longer emulate source term steps of S that use channel x for receiving, i.e., $T_2 \not\approx_{\mathsf{CMV}} [\![S]\!]^{\mathsf{CMV}^+}_{\mathsf{CMV}}$. But, since we have not yet decided whether we emulate a communication with the first or second choice on y, we also have $T_2 \not\approx_{\mathsf{CMV}} [\![S'_2]\!]^{\mathsf{CMV}^+}_{\mathsf{CMV}}$ whenever $S_2 \not\approx_{\mathsf{CMV}^+} S_4$. Indeed, if we assume that S_1, S_2, S_3, S_4 are pairwise not bisimilar, then $T_2 \not\approx_{\mathsf{CMV}} [\![S']\!]^{\mathsf{CMV}^+}_{\mathsf{CMV}}$ for all $S \longmapsto S'$, i.e., T_2 is an intermediate state.

The existence of intermediate states prevents us from using stronger versions of soundness, i.e., with $T \asymp [\![S']\!]$ instead of the requirement $T \Longmapsto_{\mathrm{T}} \asymp [\![S']\!]$ in soundness. The encoding $[\![\cdot]\!]^{\mathsf{CMV}^+}_{\mathsf{CMV}}$ needs the steps in $T \Longmapsto\approx_{\mathsf{CMV}} [\![S']\!]^{\mathsf{CMV}^+}_{\mathsf{CMV}}$ to complete the emulation of source term steps started in $[\![S]\!]^{\mathsf{CMV}^+}_{\mathsf{CMV}} \Longmapsto T$. With the soundness result we can complete the proof of [6] that $[\![\cdot]\!]^{\mathsf{CMV}^+}_{\mathsf{CMV}}$ presented in [6, § 7] is good.

Theorem 5.3 (Encoding from CMV^+ into CMV)**.** *The encoding* $[\![\cdot]\!]^{\mathsf{CMV}^+}_{\mathsf{CMV}}$ *from* CMV^+ *into* CMV *presented in [6] is good. By this encoding source terms in* CMV^+ *and their literal translations in* CMV *are related by coupled similarity.*

For the proof we take the completeness result from [6] and our soundness result in Lemma 5.2. The proof of the remaining properties is simple. That the combination of operational correspondence and barb sensitiveness induces a (weak reduction, barbed) coupled similarity that relates all source terms and their literal translations was proved in [33]. To obtain a tighter connection such as the bisimilarity, we would need the stronger version of soundness with $T \asymp [\![S']\!]$ instead of $T \Longmapsto_{\mathrm{T}} \asymp [\![S']\!]$ (see [33]).

As mentioned, a key feature of the encoding is to translate the nature of its summands, i.e., whether they are send or receive actions, into the label used by the target term. That this is possible, i.e., that the prefixes for send and receive in a choice of CMV^+ can be translated to labels in a separate choice of CMV such that the difference is not observable modulo the criteria in Definition 2.3, gives us the last piece of evidence that we need. CMV^+ does not allow to solve problems such as leader election (Theorem 3.7) that are standard problems for mixed choice; CMV^+ cannot express the synchronisation pattern $\star$ either that we associate with mixed choice (Theorem 4.5). Yet, CMV^+ can express the pattern M which is associated with *separate choice*, and is encoded by a language with only separate choice (Theorem 5.3). We conclude that choice in CMV^+ is semantically rather a separate choice.

Corollary 5.4.
The extension of CMV *given by* CMV^+ *introduces a form of separate choice rather than mixed choice.*

6 Related Work and Outlook

We conclude by discussing related work, summing up our results, and briefly discussing our next steps.

6.1 Related Work

Encodings or the proof of their absence are the main way to compare process calculi [3, 23, 10, 9, 26, 39, 24, 8, 40]. See [27] for an overview and discussion on encodings. We used this methodology to compare different variants of choice in session types.

The relevance of mixed choice for the expressive power of the π-calculus was extensively studied. An important encodability result on choices is the existence of a good encoding from the choice-free synchronous π-calculus into its asynchronous variant [4, 12], since it proves the relevance of choice. As for the separation result, [22, 10, 29] have shown that there is no good encoding from the full π-calculus, i.e., the synchronous π-calculus including mixed choice, into its asynchronous variant if an encoding should preserve the distribution of systems. Palamidessi in [21] was the first to point out that mixed choice strictly raises the expressive power of the π-calculus. Later work studies the criteria under that this separation result holds and alternative ways to prove this result: [20] studies the relevance of divergence reflection for this result and considers separate choice. [10, 23] discuss how to reprove this result if the rather strict criterion on the homomorphic translation of the parallel operator is replaced by compositionality. [26, 28] show that compositionality itself is not strong enough to replace the homomorphic translation of the parallel operator by presenting an encoding and then propose the preservation of distributability as criterion to regain the result of Palamidessi. [29] uses the more fundamental problem of breaking symmetries instead of leader election. [31] further simplifies this separation result by introducing synchronisation patterns to distinguish the languages. [32] shows that instead of the preservation of distributability or the homomorphic translation of the parallel operator also the preservation of causality can be used as criterion.

While there are a vast amount of theories [15], programming languages [1], and tools [36] of session types, as far as we know, the CMV^+-calculus is the only session π-calculus which extends external and internal choices to their mixtures with full constructs, i.e. delegation, shared (or unlimited) name passing, value passing, and recursion in its process syntax, proposes its typing system and proves type-safety. In the context of *multiparty session types* [14], there are several works that extend the original form of global types where choice is fixed (from one sender to one receiver) with more flexible forms of choices: Recent work in [17] e.g. allows the global type to specify a choice of one sender to transmit to one of several receivers. In [16] flexible choices are discussed but their well-formedness (which ensures deadlock-freedom of local types) needs to be checked by bisimuluation. These works focus on gaining expressiveness of behaviours of a set of local types (or a simple form of CCS-like processes which are equivalent to local types [17]) which correspond to *a single multiparty session*, without delegations, interleaved sessions, restrictions nor name passing.

More recently, [41] compares the expressive power of a variant of the π-calculus (with implicit matching) and the variant of CCS where the result of a synchronisation of two actions is itself an action subject to relabelling or restriction. Because of the connection between CCS-like languages and local types, it may be interesting to compare the expressiveness results in [41] with (variants of) multiparty

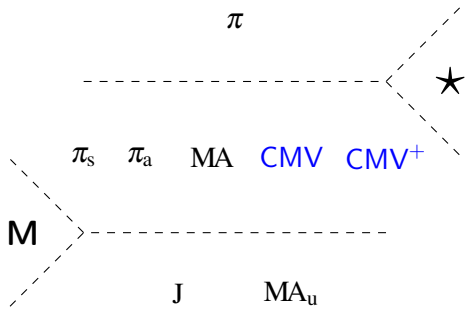


Figure 1: Hierarchy of Pi-like Calculi.

session types.

6.2 Summary and Outlook

We proved that CMV$^+$ is strictly less expressive than the π-calculus in two different ways: by showing that CMV$^+$ cannot solve leader election in symmetric networks of odd degree and that CMV$^+$ cannot express the synchronisation pattern $\star$. Then we provide the missing soundness proof for the encoding presented in [6]. From these results and the insights on the reasons of these results, we conclude that the choice primitive added to CMV in [6] is rather a separate choice and not a mixed choice at least with respect to its expressive power.

With these results we can extend the hierarchy of pi-like calculi obtained in [31, 30] by two more languages as depicted in Figure 1. This hierarchy orders languages according to their ability to express certain synchronisation patterns. At the top we have the π-calculus (π), because it can express the synchronisation pattern $\star$. In the middle are languages that can express M but not $\star$: the π-calculus with separate choice (π_s) [20], the asynchronous π-calculus without choice (π_a) [12, 4], Mobile Ambients (MA) [5], CMV, and CMV$^+$. In the bottom we have the join-calculus (J) [7] and Mobile Ambients with unique Ambient names (MA$_u$) [30], i.e., the languages that cannot express $\star$ or M. That $\pi, \pi_s, \pi_a, \text{MA}, \text{J}$, and MA$_u$ can or cannot express the respective pattern was shown in [31, 30].

Linearity as enforced by the type system of CMV/CMV$^+$ restricts the possible structures of communication protocols. In particular, the type system ensures that it is impossible to unguard two competing inputs or outputs on the same linear channel at the same time. Accordingly, it is not surprising that adding choice, even mixed choice, towards communication primitives under a type discipline that enforces linearity does not significantly increase the expressive power of the respective language (though it still might increase flexibility). However, that adding mixed choice between unrestricted communication primitives does not significantly increase the expressive power of the language, did surprise us. Unrestricted channels allow to have several in- or outputs on these channels in parallel, because the type system only ensures the absence of certain communication mismatches as e.g. that the sort of a transmitted value is as expected by the receiver; but not linearity (compare also to shared channels as e.g. in [13]). So, there is no obvious reason why the type system should limit the expressive power of unrestricted channels within a mixed choice. Indeed, it turns out that the problem lies not in the type system. In both ways to prove the separation result in § 3 and § 4 we completely ignore the type system and carry out the proof on the untyped version of the language, i.e., it is already the untyped version of CMV$^+$ that cannot express mixed choice despite a mixed-choice-like primitive. This limitation of the language definition, i.e., in its syntax and semantics, is not obvious and indeed it was very hard to spot

the problem.

We expect that adding mixed choice to the non-linear parts of other session type systems will instead significantly increase the expressive power. Accordingly, as the next step, we want to add a primitive for mixed choice between shared channels in session types such as described e.g. in [13, 42] and analyse the expressiveness of the resulting language.

Acknowledgements. The work is partially supported by EPSRC (EP/T006544/1, EP/K011715/1, EP/K034413/1, EP/L00058X/1, EP/N027833/1, EP/N028201/1, EP/T006544/1, EP/T014709/1, EP/V000462/1 and EP/X015955/1) and NCSS/EPSRC VeTSS.

References

[1] Davide Ancona, Viviana Bono, Mario Bravetti, Joana Campos, Giuseppe Castagna, Pierre-Malo Deniélou, Simon J. Gay, Nils Gesbert, Elena Giachino, Raymond Hu, Einar Broch Johnsen, Francisco Martins, Viviana Mascardi, Fabrizio Montesi, Rumyana Neykova, Nicholas Ng, Luca Padovani, Vasco T. Vasconcelos, and Nobuko Yoshida. Behavioral Types in Programming Languages. *Foundations and Trends in Programming Languages*, 3(2-3):95–230, 2016. doi:10.1561/2500000031.

[2] Benjamin Bisping, Uwe Nestmann, and Kirstin Peters. Coupled Similarity: the first 32 years. *Acta Informatica*, 57:439–463, 2019. doi:10.1007/s00236-019-00356-4.

[3] Frank S. Boer and Catuscia Palamidessi. Embedding as a tool for Language Comparison: On the CSP hierarchy. In *Proc. of CONCUR*, volume 527 of *LNCS*, pages 127–141. Springer, 1991. doi:10.1007/3-540-54430-5_85.

[4] Gérard Boudol. Asynchrony and the π-calculus (Note). Rapport de Recherche 1702, 1992. URL: https://hal.inria.fr/inria-00076939/document.

[5] Luca Cardelli and Andrew D. Gordon. Mobile ambients. *Theoretical Computer Science*, 240(1):177–213, 2000. doi:10.1016/S0304-3975(99)00231-5.

[6] Filipe Casal, Andreia Mordido, and Vasco T. Vasconcelos. Mixed sessions. *Theoretical Computer Science*, 897:23–48, 2022. doi:10.1016/j.tcs.2021.08.005.

[7] Cédric Fournet and Georges Gonthier. The Reflexive Chemical Abstract Machine and the Join-Calculus. In Jr.G̃uy Steele, editor, *Proc. of POPL*, pages 372–385. ACM, 1996. doi:10.1145/237721.237805.

[8] Yuxi Fu. Theory of Interaction. *Theoretical Computer Science*, 611:1–49, 2016. doi:10.1016/j.tcs.2015.07.043.

[9] Yuxi Fu and Hao Lu. On the expressiveness of interaction. *Theoretical Computer Science*, 411(11-13):1387–1451, 2010. doi:10.1016/j.tcs.2009.11.011.

[10] Daniele Gorla. Towards a Unified Approach to Encodability and Separation Results for Process Calculi. *Information and Computation*, 208(9):1031–1053, 2010. doi:10.1016/j.ic.2010.05.002.

[11] Kohei Honda. Types for Dyadic Interaction. In Eike Best, editor, *Proc. of CONCUR*, volume 715 of *LNCS*, pages 509–523. Springer, 1993. doi:10.1007/3-540-57208-2_35.

[12] Kohei Honda and Mario Tokoro. An Object Calculus for Asynchronous Communication. In Mario Tokoro, Oscar Nierstrasz, and Peter Wegner, editors, *Proc. of ECOOP*, volume 612 of *LNCS*, pages 133–147. Springer, 1992. doi:10.1007/BFb0057019.

[13] Kohei Honda, Vasco T. Vasconcelos, and Makoto Kubo. Language primitives and type discipline for structured communication-based programming. In *Proc. of ESOP*, volume 1381 of *LNCS*, pages 122–138. Springer, 1998. doi:10.1007/BFb0053567.

[14] Kohei Honda, Nobuko Yoshida, and Marco Carbone. Multiparty Asynchronous Session Types. *JACM*, 63:1–67, 2016. doi:10.1145/1328438.1328472.

[15] Hans Hüttel, Ivan Lanese, Vasco T. Vasconcelos, Luís Caires, Marco Carbone, Pierre-Malo Deniélou, Dimitris Mostrous, Luca Padovani, António Ravara, Emilio Tuosto, Hugo Torres Vieira, and Gianluigi Zavattaro. Foundations of Session Types and Behavioural Contracts. *ACM Computing Surveys*, 49(1):3:1–3:36, 2016. doi:10.1145/2873052.

[16] Sung-Shik Jongmans and Nobuko Yoshida. Exploring Type-Level Bisimilarity towards More Expressive Multiparty Session Types. In *Proc. of ESOP*, volume 12075 of *LNCS*, pages 251–279. Springer, 2020. doi:10.1007/978-3-030-44914-8_10.

[17] Rupak Majumdar, Madhavan Mukund, Felix Stutz, and Damien Zufferey. Generalising Projection in Asynchronous Multiparty Session Types. In Serge Haddad and Daniele Varacca, editors, *Proc. of CONCUR*, volume 203 of *LIPIcs*, pages 35:1–35:24, 2021. doi:10.4230/LIPIcs.CONCUR.2021.35.

[18] Robin Milner. *Communicating and Mobile Systems: The π-Calculus*. Cambridge University Press, 1999.

[19] Robin Milner, Joachim Parrow, and David Walker. A Calculus of Mobile Processes, Part I and II. *Information and Computation*, 100(1):1–77, 1992. doi:10.1016/0890-5401(92)90008-4.

[20] Uwe Nestmann. What is a "Good" Encoding of Guarded Choice? *Information and Computation*, 156(1-2):287–319, 2000. doi:10.1006/inco.1999.2822.

[21] Catuscia Palamidessi. Comparing the Expressive Power of the Synchronous and the Asynchronous π-calculus. In *Proc. of POPL*, pages 256–265, 1997. doi:10.1145/263699.263731.

[22] Catuscia Palamidessi. Comparing the Expressive Power of the Synchronous and the Asynchronous π-calculus. *Mathematical Structures in Computer Science*, 13(5):685–719, 2003. doi:10.1017/S0960129503004043.

[23] Joachim Parrow. Expressiveness of Process Algebras. *Electronic Notes in Theoretical Computer Science*, 209:173–186, 2008. doi:10.1016/j.entcs.2008.04.011.

[24] Joachim Parrow. General conditions for full abstraction. *Mathematical Structures in Computer Science*, 26(4):655–657, 2014. doi:10.1017/S0960129514000280.

[25] Joachim Parrow and Peter Sjödin. Multiway synchronization verified with coupled simulation. In W.R. Cleaveland, editor, *Proc. of CONCUR*, pages 518–533. Springer Berlin Heidelberg, 1992. doi:10.1007/BFb0084813.

[26] Kirstin Peters. *Translational Expressiveness*. PhD thesis, TU Berlin, 2012. URL: http://opus.kobv.de/tuberlin/volltexte/2012/3749/.

[27] Kirstin Peters. Comparing Process Calculi Using Encodings. In *Proc. of EXPRESS/SOS*, EPTCS, pages 19–38, 2019. doi:10.48550/arXiv.1908.08633.

[28] Kirstin Peters and Uwe Nestmann. Is it a "Good" Encoding of Mixed Choice? In *Proc. of FoSSaCS*, volume 7213 of *LNCS*, pages 210–224, 2012. doi:10.1007/978-3-642-28729-9_14.

[29] Kirstin Peters and Uwe Nestmann. Breaking Symmetries. *Mathematical Structures in Computer Science*, 26(6):1054–1106, 2016. doi:10.1017/S0960129514000346.

[30] Kirstin Peters and Uwe Nestmann. Distributability of Mobile Ambients. *Information and Computation*, 275:104608, 2020. doi:10.1016/j.ic.2020.104608.

[31] Kirstin Peters, Uwe Nestmann, and Ursula Goltz. On Distributability in Process Calculi. In *Proc. of ESOP*, volume 7792 of *LNCS*, pages 310–329, 2013. doi:10.1007/978-3-642-37036-6_18.

[32] Kirstin Peters, Jens-Wolfhard Schicke-Uffmann, Ursula Goltz, and Uwe Nestmann. Synchrony versus Causality in Distributed Systems. *Mathematical Structures of Computer Science*, 26:1459–1498, 2016. doi:10.1017/S0960129514000644.

[33] Kirstin Peters and Rob van Glabbeek. Analysing and Comparing Encodability Criteria. In Silvia Crafa and Daniel Gebler, editors, *Proc. of EXPRESS/SOS*, volume 190 of *EPTCS*, pages 46–60, 2015. doi:10.4204/EPTCS.190.4.

[34] Kirstin Peters and Nobuko Yoshida. On the Expressiveness of Mixed Choice Sessions (Technical Report). Technical report, 2022. doi:10.48550/arXiv.2208.07041.

[35] Gordon D. Plotkin. The origins of structural operational semantics. *Journal of Logic and Algebraic Programming*, 60:17–140, 2004. [An earlier version of this paper was published as technical report at Aarhus University in 1981.]. doi:10.1016/j.jlap.2004.03.009.

[36] António Ravara Simon Gay, editor. *Behavioural Types: from Theory to Tools*. River Publisher, 2017. URL: https://www.riverpublishers.com/research_details.php?book_id=439.

[37] Kaku Takeuchi, Kohei Honda, and Makoto Kubo. An Interaction-based Language and its Typing System. In *Proc. of PARLE*, volume 817 of *LNCS*, pages 398–413, 1994. doi:10.1007/3-540-58184-7_118.

[38] Rob van Glabbeek, Ursula Goltz, and Jens-Wolfhard Schicke. On Synchronous and Asynchronous Interaction in Distributed Systems. In *Proc. of MFCS*, volume 5162 of *LNCS*, pages 16–35, 2008. doi:10.1007/978-3-540-85238-4.

[39] Rob J. van Glabbeek. Musings on Encodings and Expressiveness. In *Proc. of EXPRESS/SOS*, volume 89 of *EPTCS*, pages 81–98, 2012. doi:10.4204/EPTCS.89.7.

[40] Rob J. van Glabbeek. A Theory of Encodings and Expressiveness (Extended Abstract). In *Proc. of FoSSaCS*, volume 10803 of *LNCS*, pages 183–202, 2018. doi:10.1007/978-3-319-89366-2_10.

[41] Rob J. van Glabbeek. Comparing the expressiveness of the π-calculus and CCS. In Ilya Sergey, editor, *Proc. of ETAPS*, volume 13240 of *LNCS*, pages 548–574. Springer, 2022. URL: https://doi.org/10.1007/978-3-030-99336-8_20, doi:10.1007/978-3-030-99336-8_20.

[42] Nobuko Yoshida and Vascos T. Vasconcelos. Language Primitives and Type Discipline for Structured Communication-Based Programming Revisited: *Two Systems for Higher-Order Session Communication*. In *Proc. of SecReT*, volume 171, pages 73–93, 2006. doi:10.1016/j.entcs.2007.02.056.