On the Expressiveness of Symmetry

Thomas Given-Wilson, Axel Legay

On the Expressiveness of Symmetry

Thomas Given-Wilson[1] and Axel Legay[1]

1 **Inria, France** {thomas.given-wilson | axel.legay}@inria.fr

Abstract

The expressiveness of communication primitives has been explored in a common framework by considering four features: *synchronism*, *arity*, *communication medium*, and *pattern-matching*. These all assume asymmetric communication between input and output primitives, however some calculi consider more *symmetric* approaches to communication such as fusion calculus and Concurrent Pattern Calculus. Symmetry can be considered either as allowing a mixture of input and output in an action or co-action, or as the unification of actions. By means of possibility/impossibility of encodings, this paper shows that: the action and co-action approach is related to or more expressive than many previously considered languages; and the unification approach is more expressive than some, but mostly unrelated to other languages.

1 Introduction

The expressiveness of process calculi based upon their choice of communication primitives has been widely explored before [34, 5, 9, 20, 13, 15]. In [20] and [15] this is detailed by examining combinations of four features, namely: *synchronism*, asynchronous vs. synchronous; *arity*, monadic vs. polyadic; *communication medium*, shared dataspaces vs. channels; and *pattern-matching*, purely binding names vs. name equality vs. intensionality. These features are able to represent many popular calculi [20, 15] including: monadic or polyadic π-calculus [30, 31, 29]; Linda [12]; significant aspects of Concurrent Pattern Calculus (CPC) [17, 18]; and Psi calculi [1].

Symmetry has been considered before in process calculi. One example, fusion calculus [35] shifts away from explicit input and output of names to instead fuse them together in a symmetric equivalence relation. Another is CPC that shifted away from input and output primitives to a single primitive that can do both input or output (and equality tests) via the unification of patterns [18].

This paper abstracts away from specific calculi in the style of [20, 15] to provide a general account of the expressiveness of *symmetric* communication primitives. Here symmetric communication does not require that input or output be associated to a particular action or co-action primitive, indeed all communication primitives can perform all possible input, output, or equality tests. This captures the spirit of both fusion calculus and CPC's interaction axioms. The complexity comes when deciding how this should be represented in an abstract calculus; there are two reasonable choices. The first is to keep the traditional difference between the two primitives involved in an interaction, perhaps calling them *action* and *co-action* in the fusion calculus style [35]. The second is to generalise from two primitives to a single primitive that interacts with others of the same kind like in CPC. The solution here is to consider both; leading to the symmetry feature having three possible instantiations. *Asymmetric* where there is explicit input (that can only contain input patterns) and output (that can only contain output terms), e.g.

$$n(\lambda x, \lambda y).P \mid \overline{n}\langle a, b\rangle.Q \quad \longmapsto \quad \{a/x, b/y\}P \mid Q \ .$$

Communication symmetric where there are two explicit primitives action and co-action that can mix input patterns and output terms, e.g.

$$n(\lambda x, b).P \mid \overline{n}\langle a, \lambda y\rangle.Q \quad \longmapsto \quad \{a/x\}P \mid \{b/y\}Q \ .$$

Fully symmetric where this is a single communication primitive that contains a single class of patterns that unify with one-another, e.g.

$$n(\lambda x \bullet b \bullet c).P \mid n(a \bullet \lambda y \bullet c).Q \quad \longmapsto \quad \{a/x\}P \mid \{b/y\}Q \ .$$

Conference title on which this volume is based on.
Editors: Billy Editor and Bill Editors; pp. 1–27
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

By extending prior work with this new dimension the original twelve calculi of [15] are here expanded to thirty-six. (Synchronism is omitted here since asynchronous calculi do not admit further behaviour after an output, but non-asymmetric languages do not have a clear output process form.) This paper details the relations between the original twelve calculi and the twenty-four new calculi.

The key results of this paper are as follows. Communication symmetric languages generalise their asymmetric but otherwise equivalent languages, except for monadic intensional languages which are equally expressive. Communication symmetric languages can be encoded into asymmetric languages as long as the communication symmetric language is monadic, or is not name-matching or intensional, this occurs when there is an additional name that can be used to represent parts of the encoding. Within communication symmetric languages expressiveness generally increases with increased arity, addition of channels, and increases in pattern-matching; with the most expressive languages being polyadic and intensional. No fully symmetric language can be encoded into a communication symmetric or asymmetric language. Within the fully symmetric languages expressiveness increases with arity, additional of channels, and pattern-matching. Fully symmetric languages do not require name-matching to be able to encode name-matching from other languages, allowing polyadic no-matching fully symmetric languages to encode all non-intensional non-fully-symmetric languages. Finally, all the intensional fully symmetric languages are equally expressive, and are the most expressive languages considered here.

These results explore two generalised approaches to symmetric communication and their expressiveness when considering other factors of communication. This captures many published calculi, as well as providing characterisations of other calculi not (yet) published. The results provide some new approaches to encoding symmetric behaviours into asymmetric languages, encoding between symmetric languages, as well as separation results related to symmetry.

The structure of the paper is as follows. Section 2 introduces the 36 calculi. Section 3 revises valid encodings. Section 4 presents a brief overview of the results. Section 5 explores new relations between and within asymmetric and communication symmetric languages. Section 6 considers fully symmetric languages and their relations to other languages. Section 7 concludes. Omitted diagrams, proofs, and discussion are in the appendices.

2 Calculi

This section defines the syntax, operational, and behavioural semantics of the calculi considered here. This relies heavily on the well-known notions developed for the π-calculus, the reference framework, and adapts them when necessary to cope with different features. With the exception of the symmetric constructs this is similar to prior definitions from [15].

Assume a countable set of names $\mathcal{N}$ ranged over by $a, b, c, \ldots$. Traditionally in π-calculus-style calculi names are used for channels, input bindings, and output data. However, here these need to be generalised to account for structure and symmetry. *Binding names*, denoted $\lambda x, \lambda y, \lambda z, \ldots$, will be used to indicate input behaviour. (Note that due to the symmetric nature of some calculi there must be a syntactic distinction between (output) names and (input) binding names, the choice of the λ for the latter here is that of CPC [18, 14], and Psi Calculi [1].) The *name-matching* patterns, denoted $m, n, o, \ldots$ are defined by $m, n ::= \lambda x \mid \ulcorner a \urcorner$ and consist of either a *binding name* λx, or a *name-match* $\ulcorner a \urcorner$ as familiar from [18, 15]. The *symmetric patterns* denoted $p, q, \ldots$ are defined by$p, q ::= a \mid m \mid p \bullet q$ The names a are as usual. The binding names λx and name-match $\ulcorner a \urcorner$ are contained in m. The *compound pattern* $p \bullet q$ combines p and q into a single pattern, and is left associative. The free names $fn(\cdot)$, binding names $bn(\cdot)$, and matched names $mn(\cdot)$ of name-matching and symmetric patterns are as expected, taking the union of sub-patterns for compound patterns. A symmetric pattern is well-formed iff all binding names are pairwise distinct. This paper only considers well-formed patterns.

$\mathcal{L}_{-,-,NO,-}$:	$INP ::= \lambda x$	$OUTP ::= a$	$ALLP ::= \lambda x \mid a$	$CH ::= a$
$\mathcal{L}_{-,-,NM,-}$:	$INP ::= m$	$OUTP ::= a$	$ALLP ::= m \mid a$	$CH ::= a$
$\mathcal{L}_{-,-,I,-}$:	$INP ::= f$	$OUTP ::= t$	$ALLP ::= p$	$CH ::= t$
$\mathcal{L}_{M,-,-,-}$:	$IN ::= INP$	$OUT ::= OUTP$	$ALL ::= ALLP$	
$\mathcal{L}_{P,-,-,-}$:	$IN ::= \widetilde{INP}$	$OUT ::= \widetilde{OUTP}$	$ALL ::= \widetilde{ALLP}$	
$\mathcal{L}_{-,D,-,A}$:	$ACT ::= (IN)$	$COACT ::= \langle OUT \rangle$		
$\mathcal{L}_{-,C,-,A}$:	$ACT ::= CH(IN)$	$COACT ::= \overline{CH}\langle OUT \rangle$		
$\mathcal{L}_{-,D,-,CS}$:	$ACT ::= (ALL)$	$COACT ::= \langle ALL \rangle$		
$\mathcal{L}_{-,C,-,CS}$:	$ACT ::= CH(ALL)$	$COACT ::= \overline{CH}\langle ALL \rangle$		
$\mathcal{L}_{-,D,-,F}$:	$ACT, COACT ::= (ALL)$			
$\mathcal{L}_{-,C,-,F}$:	$ACT, COACT ::= CH(ALL)$.			

Figure 1 Languages in this paper.

$$P \mid \mathbf{0} \equiv P \qquad P \mid Q \equiv Q \mid P \qquad P \mid (Q \mid R) \equiv (P \mid Q) \mid R \qquad P \equiv P' \quad \text{if } P =_\alpha P'$$

$$(\nu a)\mathbf{0} \equiv \mathbf{0} \qquad \textbf{if } s = t \textbf{ then } P \textbf{ else } Q \equiv P \quad s = t \qquad \textbf{if } s = t \textbf{ then } P \textbf{ else } Q \equiv Q \quad s \neq t$$

$$(\nu a)(\nu b)P \equiv (\nu b)(\nu a)P \qquad P \mid (\nu a)Q \equiv (\nu a)(P \mid Q) \quad \text{if } a \notin \mathsf{fn}(P) \qquad *P \equiv P \mid *P \, .$$

Figure 2 Structural equivalence relation.

The symmetric patterns here generalise other concepts in similar work, and also prove too general for use in some of the languages here. Thus two subsets of the symmetric patterns are defined here. The *terms* (denoted $s, t, \ldots$) are the symmetric patterns that contain no binding names or matched names. These correspond to the terms of [15], the communicable patterns of CPC, and the output structures of Psi calculi. The *intensional patterns* (denoted $f, g, \ldots$) are the symmetric patterns that contain no names (that is they consist entirely of name-matching patterns and compounds). These correspond to the intensional patterns of [15].

The (parametric) syntax for the languages is:

$$P, Q, R \quad ::= \quad \mathbf{0} \mid ACT.P \mid COACT.P \mid (\nu n)P \mid P|Q \mid \textbf{if } s = t \textbf{ then } P \textbf{ else } Q \mid *P \mid \surd \, .$$

The different languages are obtained by replacing the action ACT and co-action $COACT$ with the various definitions. The rest of the process forms as are usual.

This paper considers the possible combinations of four features for communication. As a result there exist thirty-six languages denoted by $\mathcal{L}_{\alpha,\beta,\gamma,\delta}$ where:

$\alpha =$ M for monadic data, and P for polyadic data.

$\beta =$ D for dataspace-based communication, and C for channel-based communications.

$\gamma =$ NO for no matching capability, NM for name-matching, and I for intensionality.

$\delta =$ A for asymmetric communication, CS for communication symmetric communication, and F for fully symmetric communication.

For simplicity a dash $-$ will be used when the instantiation of that feature is unimportant.

Thus the syntax of every language is obtained from the productions in Figure 1. The first 3 lines define the input, output, and all patterns (extended to tuples in polyadic languages, lines 4-5), as well as channel terms. The rest defines the languages by their action and co-action. Here the denotation $\widetilde{\cdot}$ represents a sequence of the form $\cdot_1, \cdot_2, \ldots, \cdot_n$. As usual $(\nu x)P$ and binding names λx in any form bind x in P. The corresponding notions of free and bound names of a process are as usual. Also note that alpha-conversion (denoted $=_\alpha$) is assumed in the usual manner. An action or co-action is well-formed if all binding names occur exactly once; this paper shall only consider well-formed actions and co-actions. Finally, the structural equivalence relation $\equiv$ is defined in Figure 2.

Observe that many languages here represent languages in the literature, detailed in [20, 15] for the asymmetric languages. With respect to symmetry: $\mathcal{L}_{P,C,NO,CS}$ is closest to the fusion calculus

$$\textsc{Match}(;) = (\emptyset, \emptyset)$$

$$\frac{\{p_1 \| q_1\} = (\sigma_1, \rho_1) \qquad \textsc{Match}(\widetilde{p}; \widetilde{q}) = (\sigma_2, \rho_2)}{\textsc{Match}(p_1, \widetilde{p}; q_1, \widetilde{q}) = (\sigma_1 \cup \sigma_2, \rho_1 \cup \rho_2)} \quad \begin{array}{l} p_1 \text{ is a term} \\ q_1 \text{ is an intensional pattern} \end{array}$$

$$\frac{\{p_1 \| q_1\} = (\sigma_1, \rho_1) \qquad \textsc{Match}(\widetilde{p}; \widetilde{q}) = (\sigma_2, \rho_2)}{\textsc{Match}(p_1, \widetilde{p}; q_1, \widetilde{q}) = (\sigma_1 \cup \sigma_2, \rho_1 \cup \rho_2)} \quad \begin{array}{l} p_1 \text{ is an intensional pattern} \\ q_1 \text{ is a term} \end{array}$$

Figure 3 The poly-match rules.

[35] although the scope of binding in communication is different; $\mathcal{L}_{M,D,I,F}$ corresponds to CPC; and $\mathcal{L}_{M,D,I,CS}$, $\mathcal{L}_{M,C,I,CS}$, and $\mathcal{L}_{M,C,I,F}$ to variants of CPC [13].

▶ Remark 1. Most of the languages can be ordered; in particular $\mathcal{L}_{\alpha_1,\beta_1,\gamma_1,\delta_1}$ can be encoded into $\mathcal{L}_{\alpha_2,\beta_2,\gamma_2,\delta_2}$ if it holds that $\alpha_1 \leq \alpha_2$ and $\beta_1 \leq \beta_2$ and $\gamma_1 \leq \gamma_2$ and $\delta_1 \leq \delta_2$, where $\leq$ is the least reflexive relation satisfying $M \leq P$ and $D \leq C$ and $NO \leq NM \leq I$ and $A \leq CS$. This can be understood as the lesser language variation being a special case of the greater language. Monadic communication is polyadic communication with all tuples of arity one. Dataspace-based communication is channel-based communication with all k-ary tuples communicating with channel name k. All name-matching communication is intensional communication without any compounds, and no-matching capability communication is both without any compounds and with only names or only binding names in patterns. Lastly, asymmetric communication is communication-symmetric with only input patterns in actions, and only output patterns in co-actions; and communication-symmetric languages are fully-symmetric languages with restrictions upon the unification (this does not induce $\leq$, see Section 6).

The operational semantics of the languages is given here via reductions as in [15]. An alternative style is via a *labelled transition system* (LTS) such as [20]. Here the reduction based style is chosen for simplicity. However, the LTS style can be used for intensional and symmetric languages [1, 13, 16], and indeed captures the languages here [16].

Substitutions, denoted $\sigma, \rho, \ldots$, for a language $\mathcal{L}_{-,-,\gamma,-}$ are finite mappings from names to: names ($\gamma \neq I$), or terms ($\gamma = I$). The application of a substitution σ to a pattern p is defined in the obvious manner, applying to sub-patterns of compounds, and on the understanding that $\ulcorner(s \bullet t)\urcorner \stackrel{\text{def}}{=} \ulcorner s \urcorner \bullet \ulcorner t \urcorner$. Given a substitution σ and a process P, denote with σP the capture-avoiding application of σ to P as usual. As usual capture can be avoided by exploiting α-equivalence, assumed due to [39, 2].

Interaction between processes is handled by unification of patterns. The core unification can be used for all languages as defined by the *unify* rule $\{p \| q\}$ of patterns p and q to create two substitutions σ and ρ whose domains are the binding names of p and q, respectively. This is defined as follows:

$$\begin{array}{rcll} \{a \| a\} = \{a \| \ulcorner a \urcorner\} = \{\ulcorner a \urcorner \| a\} = \{\ulcorner a \urcorner \| \ulcorner a \urcorner\} & \stackrel{\text{def}}{=} & (\{\}, \{\}) & \\ \{\lambda x \| t\} & \stackrel{\text{def}}{=} & (\{t/x\}, \{\}) & \text{if } t \text{ is a term} \\ \{s \| \lambda x\} & \stackrel{\text{def}}{=} & (\{\}, \{s/x\}) & \text{if } s \text{ is a term} \\ \{p_1 \bullet p_2 \| q_1 \bullet q_2\} & \stackrel{\text{def}}{=} & (\sigma_1 \cup \sigma_2\,,\, \rho_1 \cup \rho_2) & \text{if } \{p_i \| q_i\} = (\sigma_i, \rho_i) \text{ for } i \in \{1, 2\} \\ \{p \| q\} & & \text{undefined} & \text{otherwise.} \end{array}$$

Names and name-matches unify if they are for the same name. A binding name unifies with a term to produce a binding of the name to that term. Two compounds unify if their components unify; the resulting substitutions are the unions of those produced by unifying the components. Otherwise the unification is undefined (impossible). Note that the substitutions being combined have disjoint domain due to well-formedness of patterns, and this holds for the following two rules also.

The asymmetric and communication symmetric languages exploit the *poly-match* rule $\textsc{Match}(\widetilde{p}; \widetilde{q})$ that determines the matches of two sequences of patterns $\widetilde{p}$ and $\widetilde{q}$ to produce a pair of substitutions, as defined in Figure 3. The empty sequence matches with itself to produce empty substitutions.

Otherwise when there are sequences $p_1, \widetilde{p}$ and $q_1, \widetilde{q}$ where p_1 is a term and q_1 is an intensional pattern (or vice versa) then they are unified $\{p_1 \| q_1\}$ and the remaining sequences use the poly-match rule. If both are defined and yield substitutions, the union of substitutions is yielded. Otherwise the poly-match is undefined.

The fully symmetric languages use the *poly-unify* rule $\textsc{Unify}(\widetilde{p}; \widetilde{q})$ that is the same as the poly-match rule (without the side conditions) as shown below:

$$\textsc{Unify}(;) = (\emptyset, \emptyset) \qquad \frac{\{p_1 \| q_1\} = (\sigma_1, \rho_1) \qquad \textsc{Unify}(\widetilde{p}; \widetilde{q}) = (\sigma_2, \rho_2)}{\textsc{Unify}(p_1, \widetilde{p}; q_1, \widetilde{q}) = (\sigma_1 \cup \sigma_2, \rho_1 \cup \rho_2)}.$$

Interaction is now defined by the following two axioms, first:

$$\overline{s}\langle \widetilde{p} \rangle.P \mid s(\widetilde{q}).Q \quad \longmapsto \quad (\sigma P) \mid (\rho Q) \qquad \textsc{Match}(\widetilde{p}; \widetilde{q}) = (\sigma, \rho)$$

for the asymmetric and communication symmetric languages, and second:

$$s(\widetilde{p}).P \mid s(\widetilde{q}).Q \quad \longmapsto \quad (\sigma P) \mid (\rho Q) \qquad \textsc{Unify}(\widetilde{p}; \widetilde{q}) = (\sigma, \rho)$$

for the fully symmetric languages, and where the s's are omitted in both for the dataspace-based languages. Both axioms state that when the the symmetric patterns $\widetilde{p}$ and $\widetilde{q}$ poly-match or poly-unify, respectively, (and in the channel-based setting the input and output are along the same channel) to yield the substitutions σ and ρ, they reduce to σ applied to P in parallel with ρ applied to Q. The reduction relation $\longmapsto$ also includes the following:

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

with $\Longmapsto$ denoting the reflexive, transitive closure of $\longmapsto$. Lastly, for each language let $\simeq$ denote a reduction-sensitive reference behavioural equivalence, defined in [20, 16, 15] or derived from [16].

3 Encodings

This section recalls the definition of valid encodings and results for formally relating process calculi [22]. The validity of such criteria in developing expressiveness studies emerges from the various works [20, 21, 22], that have also recently inspired similar works [27, 40, 15].

An *encoding* of a language $\mathcal{L}_1$ into another language $\mathcal{L}_2$ is a pair $(\llbracket \cdot \rrbracket, \varphi_{\llbracket \rrbracket})$ where $\llbracket \cdot \rrbracket$ translates every $\mathcal{L}_1$-process into an $\mathcal{L}_2$-process and $\varphi_{\llbracket \rrbracket}$ maps every name into a tuple of k names for $k > 0$. The translation $\llbracket \cdot \rrbracket$ turns every term of the source language into a term of the target. Now consider only encodings that satisfy the following properties. Let a *k-ary context* $C(\__1; \ldots; \__k)$ be a term where k occurrences of $\mathbf{0}$ are linearly replaced by the holes $\{\__1; \ldots; \__k\}$. Moreover, denote with $\longmapsto_i$ and $\Longmapsto_i$ the relations $\longmapsto$ and $\Longmapsto$ in language $\mathcal{L}_i$; denote with $\longmapsto_i^\omega$ an infinite sequence of reductions in $\mathcal{L}_i$. Also, let $\simeq_i$ denote the reference behavioural equivalence for language $\mathcal{L}_i$. Further, let $P \Downarrow_i$ mean that there exists P' such that $P \Longmapsto_i P'$ and $P' \equiv P'' \mid \surd$, for some P''.

▶ **Definition 2** (Valid Encoding)**.** An encoding $(\llbracket \cdot \rrbracket, \varphi_{\llbracket \rrbracket})$ of $\mathcal{L}_1$ into $\mathcal{L}_2$ is *valid* if it satisfies the following five properties:

1. *Compositionality:* for every k-ary operator op of $\mathcal{L}_1$ and for every subset of names N, there exists a k-ary context $C^N_{\mathsf{op}}(\__1; \ldots; \__k)$ of $\mathcal{L}_2$ such that, for all $S_1, \ldots, S_k$ with $\mathsf{fn}(S_1, \ldots, S_k) = N$, it holds that $\llbracket \mathsf{op}(S_1, \ldots, S_k) \rrbracket = C^N_{\mathsf{op}}(\llbracket S_1 \rrbracket; \ldots; \llbracket S_k \rrbracket)$.
2. *Name invariance:* for every S and substitution σ, it holds that $\llbracket \sigma S \rrbracket = \sigma' \llbracket S \rrbracket$ if σ is injective and $\llbracket \sigma S \rrbracket \simeq_2 \sigma' \llbracket S \rrbracket$ otherwise where σ' is such that $\varphi_{\llbracket \rrbracket}(\sigma(a)) = \sigma'(\varphi_{\llbracket \rrbracket}(a))$ for every name a.
3. *Operational correspondence:* for all $S \Longmapsto_1 S'$, it holds that $\llbracket S \rrbracket \Longmapsto_2 \simeq_2 \llbracket S' \rrbracket$; and for all $\llbracket S \rrbracket \Longmapsto_2 T$, there exists S' such that $S \Longmapsto_1 S'$ and $T \Longmapsto_2 \simeq_2 \llbracket S' \rrbracket$.

4. *Divergence reflection:* for every S such that $[\![S]\!] \longmapsto_2^\omega$, it holds that $S \longmapsto_1^\omega$.
5. *Success sensitiveness:* for every S, it holds that $S \Downarrow_1$ if and only if $[\![S]\!] \Downarrow_2$.

▶ Proposition 3 (5.5 from [22]). Let $[\![\cdot]\!]$ be a valid encoding; then, $S \mapsto\!\!\!\!\!\rightarrow_1$ implies that $[\![S]\!] \mapsto\!\!\!\!\!\rightarrow_2$.

▶ **Theorem 4** (5.8 from [22]). *Assume there exists a $\mathcal{L}_1$-process S such that $S \mapsto\!\!\!\!\!\rightarrow_1$ and $S \Downarrow$ and $S \mid S \Downarrow$; moreover assume that every $\mathcal{L}_2$-process T that does not reduce is such that $T \mid T \mapsto\!\!\!\!\!\rightarrow_2$. Then there cannot exist a valid encoding $[\![\cdot]\!]$ from $\mathcal{L}_1$ to $\mathcal{L}_2$.*

▶ **Definition 5** (Matching Degree from [22]). The *matching degree* of a language $\mathcal{L}$, written $\text{MD}(\mathcal{L})$, is the least upper bound on the number of names that must be matched to yield a reduction in $\mathcal{L}$.

▶ **Theorem 6** (5.9 from [22]). *If $\text{MD}(\mathcal{L}_1) > \text{MD}(\mathcal{L}_2)$ then there is no valid encoding of $\mathcal{L}_1$ into $\mathcal{L}_2$.*

4 Overview of Results

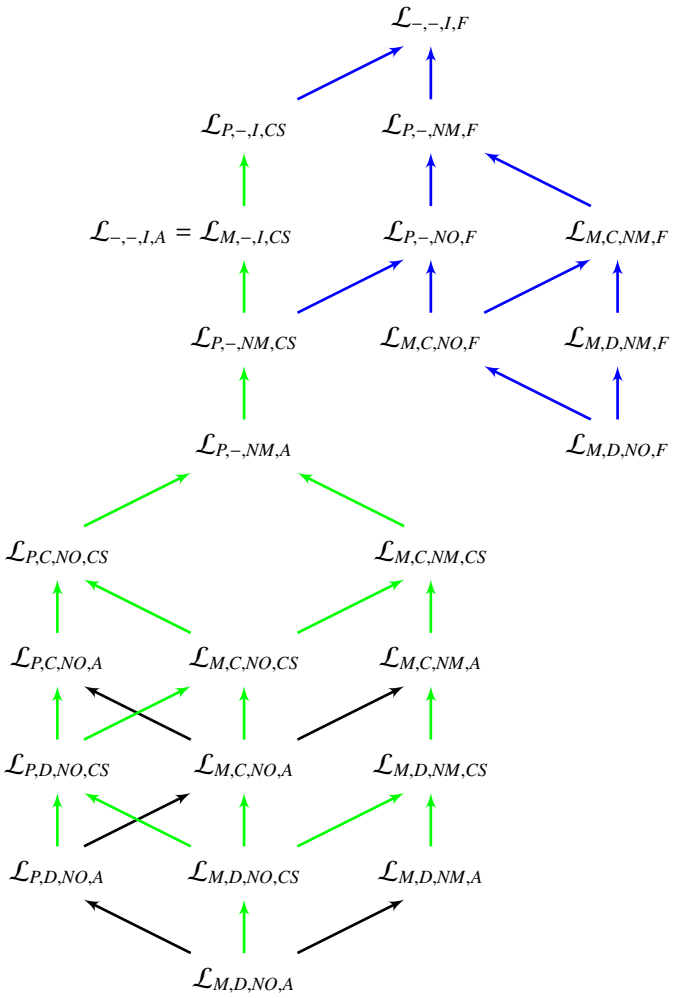

Figure 4 Relations between all languages

A diagram illustrating the results can be seen in Figure 4. Arrows show increased expressive power; black are from prior work, green arrows from Section 5, and blue arrows from Section 6. The lack of an arrow indicates no possible encoding in either direction (e.g. between $\mathcal{L}_{P,-,I,CS}$ and $\mathcal{L}_{P,-,NM,F}$). Transitive relations are omitted (e.g. $\mathcal{L}_{P,-,NM,A}$ to $\mathcal{L}_{P,-,NO,F}$).

5 Asymmetry and Communication Symmetry

Communication symmetry is strictly a generalisation of asymmetric communication; that is every language $\mathcal{L}_{\alpha,\beta,\gamma,A}$ is a sub-language of, and thus trivially encoded by, $\mathcal{L}_{\alpha,\beta,\gamma,CS}$. The rest of this section details relations between asymmetric and communication symmetric languages.

5.1 Adding Symmetry to Monadic Non-Intensional Languages

For the monadic non-intensional languages changing from asymmetric to communication symmetric alone is always an increase in expressive power. That is, $\mathcal{L}_{M,\beta,\gamma,CS}$ is always more expressive than $\mathcal{L}_{M,\beta,\gamma,A}$ when $\gamma \neq I$.

▶ **Theorem 7.** *There exists no valid encoding of $\mathcal{L}_{M,D,NO,CS}$ into $\mathcal{L}_{M,D,NO,A}$.*

▶ **Theorem 8.** *There exists no valid encoding of $\mathcal{L}_{M,D,NM,CS}$ into $\mathcal{L}_{M,D,NM,A}$.*

▶ **Theorem 9.** *There exists no valid encoding of $\mathcal{L}_{M,C,NO,CS}$ into $\mathcal{L}_{M,C,NO,A}$.*

▶ **Theorem 10.** *There exists no valid encoding of $\mathcal{L}_{M,C,NM,CS}$ into $\mathcal{L}_{M,C,NM,A}$.*

Within the monadic non-intensional communication symmetric languages the usual diamond of relations exists where adding channel-based communication or name-matching are both increases in expressive power. All results except the following are obtained via the relative matching degrees.

▶ **Theorem 11.** *The languages $\mathcal{L}_{M,D,NM,CS}$ and $\mathcal{L}_{M,C,NO,CS}$ are unrelated.*

5.2 Encoding Communication Symmetry in Asymmetry

A communication symmetric language $\mathcal{L}_1$ can be encoded into an asymmetric language $\mathcal{L}_2$ if the matching degree of $\mathcal{L}_1$ is bounded, and $\mathcal{L}_2$ has a greater matching degree and is either polyadic or channel-based. The key idea is that a single name is sufficient to represent the shape of the encoded action or co-action, and so can ensure correct encoded interactions. Observe that in every case the reverse encoding is proved impossible easily via the matching degree and Theorem 6.

The clearest illustration of this when the source language is monadic is the following encoding from $\mathcal{L}_{M,D,NO,CS}$ into $\mathcal{L}_{M,C,NO,A}$. Consider the translation $[\![\cdot]\!]$ that is homeomorphic on all forms except for the action and co-action, and exploits two reserved names ia and ic (such reserved names can be ensured by the renaming policy [22]) that are translated as follows:

$$[\![(p).P]\!] \stackrel{\text{def}}{=} \begin{cases} \overline{\mathsf{ic}}\langle a\rangle.[\![P]\!] & p = a \\ \mathsf{ia}(\lambda x).[\![P]\!] & p = \lambda x \end{cases} \qquad [\![\langle p\rangle.P]\!] \stackrel{\text{def}}{=} \begin{cases} \overline{\mathsf{ia}}\langle a\rangle.[\![P]\!] & p = a \\ \mathsf{ic}(\lambda x).[\![P]\!] & p = \lambda x \,. \end{cases}$$

Here the channel name indicates the origin of the input, ia for input on action, and ic for co-action.

▶ **Theorem 12.** *The encoding from $\mathcal{L}_{M,D,NO,CS}$ into $\mathcal{L}_{M,C,NO,A}$ is valid.*

▶ **Theorem 13.** *There exists no valid encoding from $\mathcal{L}_{M,C,NO,A}$ into $\mathcal{L}_{M,D,NO,CS}$.*

The following theorem is achieved in the same manner by considering name matches $\ulcorner a \urcorner$ to also be inputs, i.e. $[\![(\ulcorner a \urcorner).P]\!] \stackrel{\text{def}}{=} \mathsf{ia}(\ulcorner a \urcorner).[\![P]\!]$ and $[\![\langle \ulcorner a \urcorner\rangle.P]\!] \stackrel{\text{def}}{=} \mathsf{ic}(\ulcorner a \urcorner).[\![P]\!]$.

▶ **Theorem 14.** *The encoding from $\mathcal{L}_{M,D,NM,CS}$ into $\mathcal{L}_{M,C,NM,A}$ is valid.*

▶ **Theorem 15.** *There exists no valid encoding from $\mathcal{L}_{M,C,NM,A}$ into $\mathcal{L}_{M,D,NM,CS}$.*

The following result that the languages $\mathcal{L}_{M,C,NM,CS}$ can be encoded into $\mathcal{L}_{P,-,NM,A}$ follows from the above result, where instead of the reserved names being used as a channel they are simply added as another part of the polyadic input or output, with appropriate name-matching on the input.

▶ **Theorem 16.** *There exist valid encodings from $\mathcal{L}_{M,C,NM,CS}$ into $\mathcal{L}_{P,-,NM,A}$.*

▶ **Theorem 17.** *There exists no valid encoding from $\mathcal{L}_{P,-,NM,A}$ into $\mathcal{L}_{M,C,NM,CS}$.*

A similar but more complex technique can be used to encode polyadic no-matching communication symmetric languages into asymmetric languages. This is illustrated by the following encoding from $\mathcal{L}_{P,D,NO,CS}$ into $\mathcal{L}_{P,C,NO,A}$.

The encoding exploits a binary representation of the structure of an action or co-action. To this end define the *binary representation function* $\textsc{Bin}(\widetilde{\cdot})$ that converts a sequence of binding names and names into a bit-string of ones $\mathtt{1}$ and zeros $\mathtt{0}$, respectively. For example, $\textsc{Bin}(\lambda x, a) = \mathtt{10}$ and $\textsc{Bin}(a, \lambda x) = \mathtt{01}$. Also define the *complement* (or bitwise not) $\textsc{Not}(\cdot)$ of bit-strings in the usual manner. Given a sequence of binding names and names $\widetilde{p}$, the order preserving sequences of the binding names $\textsc{Bn}(\widetilde{p})$, and names $\textsc{Nm}(\widetilde{p})$ are defined in the obvious manner. Now consider the translation $[\![\cdot]\!]$ that is homeomorphic on all forms except the action and co-action (and exploits a reserved name rn) that are translated as follows:

$$\begin{aligned} [\![(\widetilde{p}).P]\!] &\stackrel{\text{def}}{=} a(\lambda \mathsf{rn}, \textsc{Bn}(\widetilde{p})).\overline{\mathsf{rn}}\langle \textsc{Nm}(\widetilde{p})\rangle.[\![P]\!] && a = \textsc{Bin}(\widetilde{p}) \\ [\![\langle \widetilde{p}\rangle.P]\!] &\stackrel{\text{def}}{=} (\nu \mathsf{rn})\overline{a}\langle \mathsf{rn}, \textsc{Nm}(\widetilde{p})\rangle.\mathsf{rn}(\textsc{Bn}(\widetilde{p})).[\![P]\!] && a = \textsc{Not}(\textsc{Bin}(\widetilde{p})) \,. \end{aligned}$$

The idea is that the translated action and co-action match on the channel name that is the bit-string representation of their order of binding names and names. If they match the input performs all the action's bindings as well as an additional name (bound to) rn. The rôles are then reversed with the encoded action sending all its names using the private channel rn.

▶ **Theorem 18.** *The encoding from $\mathcal{L}_{P,D,NO,CS}$ into $\mathcal{L}_{P,C,NO,A}$ is valid.*

▶ **Theorem 19.** *There exists no valid encoding from $\mathcal{L}_{P,C,NO,A}$ into $\mathcal{L}_{P,D,NO,CS}$.*

Building on this and the equivalence between the languages $\mathcal{L}_{P,-,NM,A}$ [20] conclude that $\mathcal{L}_{P,-,NM,A}$ are able to encode all languages $\mathcal{L}_{M,-,\gamma,CS}$ where $\gamma \neq I$ and $\mathcal{L}_{P,-,NO,CS}$ and $\mathcal{L}_{-,-,\gamma,A}$ where $\gamma \neq I$.

5.3 Other Relations With Bounded Matching Degree

This section considers other relations between languages equally or less expressive than $\mathcal{L}_{P,-,NM,A}$.

$\mathcal{L}_{P,\beta,NO,CS}$ are more expressive than $\mathcal{L}_{M,\beta,NO,CS}$: Clearly $\mathcal{L}_{M,\beta,NO,CS}$ is a sub-language of $\mathcal{L}_{P,\beta,NO,CS}$ for any β and so can be validly encoded in it. The following proves the separation result.

▶ **Theorem 20.** *There exists no valid encoding of $\mathcal{L}_{P,D,NO,CS}$ into $\mathcal{L}_{M,D,NO,CS}$.*

▶ **Theorem 21.** *There exists no valid encoding of $\mathcal{L}_{P,C,NO,CS}$ into $\mathcal{L}_{M,C,NO,CS}$.*

$\mathcal{L}_{P,D,NO,CS}$ is more expressive than $\mathcal{L}_{P,D,NO,A}$: Clearly $\mathcal{L}_{P,D,NO,A}$ is a sub-language of $\mathcal{L}_{P,D,NO,CS}$ and can be validly encoded in it. The following proves an increase in expressiveness.

▶ **Theorem 22.** *There exists no valid encoding of $\mathcal{L}_{P,D,NO,CS}$ into $\mathcal{L}_{P,D,NO,A}$.*

5.4 Equivalent Languages with Unbounded Matching Degree

Once the matching degree is unbounded several languages become equivalent in expressiveness. This section formalises equivalences between groups of languages that have unbounded matching degree.

The intensional asymmetric languages have equivalent expressiveness to one another and to the monadic communication symmetric languages. The equivalence between all the asymmetric languages comes from Theorem 6.5 of [15]. Now consider the languages $\mathcal{L}_{M,D,I,CS}$ and $\mathcal{L}_{M,C,I,CS}$. Clearly there is a trivial valid encoding of $\mathcal{L}_{M,D,I,CS}$ into $\mathcal{L}_{M,C,I,CS}$. In the other direction take the encoding $[\![\cdot]\!]$ that is the homeomorphic on all processes except the action and co-action that are encoded as follows (exploiting reserved names as usual):

$$\left.\begin{array}{l} [\![\, p(q).P \,]\!] \stackrel{\text{def}}{=} \langle \mathsf{ic} \bullet p \bullet q \rangle.[\![\, P \,]\!] \\ [\![\, \overline{p}\langle q \rangle.P \,]\!] \stackrel{\text{def}}{=} \langle \mathsf{ia} \bullet p \bullet q \rangle.[\![\, P \,]\!] \end{array}\right\} \; q \text{ is a term} \qquad \left.\begin{array}{l} [\![\, p(q).P \,]\!] \stackrel{\text{def}}{=} (\mathsf{ia} \bullet \ulcorner p \urcorner \bullet q).[\![\, P \,]\!] \\ [\![\, \overline{p}\langle q \rangle.P \,]\!] \stackrel{\text{def}}{=} (\mathsf{ic} \bullet \ulcorner p \urcorner \bullet q).[\![\, P \,]\!] \end{array}\right\} \; q \text{ is an intensional pattern.}$$

The translation is straightforward by compounding the channel pattern p with the term or intensional pattern q, converting to maintain being either a term or intensional pattern as required.

▶ **Theorem 23.** *The encoding from $\mathcal{L}_{M,C,I,CS}$ into $\mathcal{L}_{M,D,I,CS}$ is valid.*

The following completes the equivalences. Since all the languages $\mathcal{L}_{-,-,I,A}$ are equally expressive [15] and since the languages $\mathcal{L}_{M,-,I,CS}$ are equally expressive by Theorem 23 it is sufficient to consider specific examples from either group. The encodings from $\mathcal{L}_{-,-,I,A}$ into $\mathcal{L}_{M,-,I,CS}$ follow by $\mathcal{L}_{M,D,I,A}$ being a sub-language of $\mathcal{L}_{M,D,I,CS}$. The other direction follows from the following theorem that completes the equivalences, via a straightforward adaption of the proof of Theorem 12.

▶ **Theorem 24.** *There is a valid encoding from $\mathcal{L}_{M,D,I,CS}$ into $\mathcal{L}_{M,C,I,A}$.*

$\mathcal{L}_{P,C,NM,CS}$ is equally expressive as $\mathcal{L}_{P,D,NM,CS}$: Clearly $\mathcal{L}_{P,D,NM,CS}$ can be trivially validly encoded into $\mathcal{L}_{P,C,NM,CS}$. In the other direction consider the translation $[\![\cdot]\!]$ from $\mathcal{L}_{P,C,NM,CS}$ into $\mathcal{L}_{P,D,NM,CS}$ that is the homeomorphic on all forms except the action and co-action which are as follows:

$$[\![a(\widetilde{p}).P]\!] \stackrel{\text{def}}{=} (\ulcorner a \urcorner, \widetilde{p}).[\![P]\!] \qquad [\![\overline{a}\langle\widetilde{p}\rangle.P]\!] \stackrel{\text{def}}{=} \langle a, \widetilde{p}\rangle.[\![P]\!] .$$

The translation is the standard approach [20, 15] for polyadic name-matching languages that increases the arity by one and shifts the channel name to the first position (with an appropriate pattern/term).

▶ **Theorem 25.** *The encoding from $\mathcal{L}_{P,C,NM,CS}$ into $\mathcal{L}_{P,D,NM,CS}$ is valid.*

The languages $\mathcal{L}_{P,D,I,CS}$ and $\mathcal{L}_{P,C,I,CS}$ are equally expressive: Clearly there is a trivial valid encoding of $\mathcal{L}_{P,D,I,CS}$ into $\mathcal{L}_{P,C,I,CS}$. The following shows the reverse yielding equivalent expressiveness.

▶ **Theorem 26.** *The exists a valid encoding from $\mathcal{L}_{P,C,I,CS}$ into $\mathcal{L}_{P,D,I,CS}$.*

5.5 Concluding Relations

$\mathcal{L}_{P,-,I,CS}$ are more expressive than $\mathcal{L}_{P,-,NM,CS}$: By Theorems 25 & 26 being channel-based or dataspace-based is unimportant. The languages $\mathcal{L}_{P,-,NM,CS}$ are sub-languages of $\mathcal{L}_{P,-,I,CS}$. The following result proves increased expressiveness.

▶ **Theorem 27.** *There exists no valid encoding of $\mathcal{L}_{P,-,I,CS}$ into $\mathcal{L}_{P,-,NM,CS}$.*

$\mathcal{L}_{M,-,I,CS}$ are unrelated to $\mathcal{L}_{P,-,NM,CS}$: Note the groups of languages can be treated equivalently due to Theorems 24 & 25. The proof techniques below are the same as Theorems 27 & 20, respectively.

▶ **Theorem 28.** *There exists no valid encoding from $\mathcal{L}_{M,-,I,CS}$ into $\mathcal{L}_{P,-,NM,CS}$.*

▶ **Theorem 29.** *There exists no valid encoding from $\mathcal{L}_{P,-,NM,CS}$ into $\mathcal{L}_{M,-,I,CS}$.*

$\mathcal{L}_{P,-,NM,CS}$ are more expressive than $\mathcal{L}_{P,-,NM,A}$: By Theorem 25 and Proposition 4.1 of [20] in all languages considered here being channel-based or dataspace-based is immaterial to the expressive power. The languages $\mathcal{L}_{P,-,NM,A}$ are sub-languages of $\mathcal{L}_{P,-,NM,CS}$ and so their expressiveness is included. The reverse is prevented by the following result.

▶ **Theorem 30.** *There exists no valid encoding from $\mathcal{L}_{P,-,NM,CS}$ into $\mathcal{L}_{P,-,NM,A}$.*

$\mathcal{L}_{P,-,I,CS}$ are more expressive than $\mathcal{L}_{M,-,I,CS}$: By Theorems 24 & 26 in all languages considered here being channel-based or dataspace-based is immaterial to the expressive power. The languages $\mathcal{L}_{M,-,I,CS}$ are sub-languages of $\mathcal{L}_{P,-,I,CS}$ and so their expressiveness is included. The reverse separation result is the same technique as Theorem 30.

▶ **Theorem 31.** *There exists no valid encoding from $\mathcal{L}_{P,-,I,CS}$ into $\mathcal{L}_{M,-,I,CS}$.*

6 Fully Symmetric Communication

6.1 Full Symmetry Cannot be Simulated

The following result shows that no fully symmetric language can be encoded into an asymmetric or communication symmetric language. The key to the result is a *self recognising* process, defined to be $(a).\surd$ for the dataspace-based languages and $a(a).\surd$ for the channel-based languages. Observe that $S = (a).\surd$ has the behaviour $S \mid S \longmapsto\!\!\!\!\Downarrow$ but $S \not\longmapsto$ and $S \not\Downarrow$. This can be exploited since no asymmetric or communication symmetric process can reduce in parallel with itself unless it reduces alone. The self recognising process can therefore be used to yield the following result.

▶ **Theorem 32.** *There exists no valid encoding of a fully symmetric language $\mathcal{L}_{-,-,-,F}$ into any asymmetric or communication symmetric language $\mathcal{L}_{-,-,-,\delta}$ $\delta \neq F$.*

The above result can be used to prove a separation result from any fully symmetric language to a non-fully symmetric language, and so these results shall be omitted from the rest of the paper.

6.2 On Monadic Non-Intensional Fully Symmetric Languages

All the monadic non-intensional fully symmetric languages are unrelated to any non-symmetric language. Similar to the asymmetric or communication symmetric languages, these 4 form a diamond where expressiveness is increased by adding channel-based communication or pattern-matching.

The shift to fully symmetric communication leads to $\mathcal{L}_{M,D,NO,F}$ being unrelated to any other monadic, dataspace-based, non-matching language $\mathcal{L}_{M,D,NO,-}$.

▶ **Theorem 33.** *There exists no valid encoding from $\mathcal{L}_{M,D,NO,\delta}$ where $\delta \neq F$ into $\mathcal{L}_{M,D,NO,F}$.*

▶ **Theorem 34.** *There exists no valid encoding from $\mathcal{L}_{M,-,\gamma,\delta}$ into $\mathcal{L}_{M,-,\gamma,F}$ where $\gamma \leq NM$ and $\delta \neq F$.*

The relations between the monadic non-intensional fully symmetric languages are as usual, although the usual proof techniques do not always hold. In particular, no-matching fully symmetric languages still have non-zero matching degree, so separation results that rely on matching degree alone no longer hold. This is illustrated below.

Clearly $\mathcal{L}_{M,D,NO,F}$ is a sub-language of $\mathcal{L}_{M,D,NM,F}$ and can be validly encoded in it. The following proves the separation result required to indicate an increase in expressiveness.

▶ **Theorem 35.** *There exists no valid encoding of $\mathcal{L}_{M,D,NM,F}$ into $\mathcal{L}_{M,D,NO,F}$.*

The same approach can be used again when both languages are channel-based. Clearly $\mathcal{L}_{M,C,NO,F}$ is a sub-language of $\mathcal{L}_{M,C,NM,F}$ and can be validly encoded in it. The following proves the separation result required to indicate an increase in expressiveness.

▶ **Theorem 36.** *There exists no valid encoding of $\mathcal{L}_{M,C,NM,F}$ into $\mathcal{L}_{M,C,NO,F}$.*

When one language is dataspace-based and the other channel-based then the matching degree suffices. Clearly $\mathcal{L}_{M,D,NO,F}$ and $\mathcal{L}_{M,D,NM,F}$ can be trivially validly encoded into $\mathcal{L}_{M,C,NO,F}$ and $\mathcal{L}_{M,C,NM,F}$ respectively. The following results prove increase in expressiveness via matching degree.

▶ **Theorem 37.** *There exists no valid encoding of $\mathcal{L}_{M,C,NO,F}$ into $\mathcal{L}_{M,D,NO,F}$.*

▶ **Theorem 38.** *There exists no valid encoding of $\mathcal{L}_{M,C,NM,F}$ into $\mathcal{L}_{M,D,NM,F}$.*

▶ **Theorem 39.** *The languages $\mathcal{L}_{M,D,NM,F}$ and $\mathcal{L}_{M,C,NO,F}$ are unrelated.*

6.3 Equally Expressive Fully Symmetric Languages

Once the matching degree is unbounded there is no difference in expressiveness between dataspace-based and channel-based communication for fully symmetric languages. Further, all the intensional fully symmetric languages have equal expressive power. The rest of this section formalises this.

For the polyadic languages it is straightforward to represent channel-based communication by shifting the channel to the first position of a dataspace-based encoding. There is a trivial valid encoding of $\mathcal{L}_{P,D,NO,F}$ into $\mathcal{L}_{P,C,NO,F}$. The following shows equivalence via the reverse encoding $\mathcal{L}_{P,C,NO,F}$ into $\mathcal{L}_{P,D,NO,F}$. Consider the encoding $[\![\cdot]\!]$ from $\mathcal{L}_{P,C,NO,F}$ to $\mathcal{L}_{P,D,NO,F}$ that is the homeomorphic on all forms except the action that is $[\![a(\widetilde{p}).P]\!] \stackrel{\text{def}}{=} (a, \widetilde{p}).[\![P]\!]$. Here the channel name is merely shifted to the first position in the sequence of patterns of the polyadic action.

▶ **Theorem 40.** *The encoding from $\mathcal{L}_{P,C,NO,F}$ into $\mathcal{L}_{P,D,NO,F}$ is valid.*

This result may at first appear unexpected since in the asymmetric and communication-symmetric languages $\mathcal{L}_{P,C,NO,\delta}$ ($\delta \neq F$) have matching degree 1 while $\mathcal{L}_{P,D,NO,\delta}$ ($\delta \neq F$) have matching degree 0. However, this does not hold for fully symmetric languages as due to the poly-unify rule their matching degree directly relates to their arity.

The equivalence between $\mathcal{L}_{P,D,NM,F}$ and $\mathcal{L}_{P,C,NM,F}$ can be shown in the same manner.

▶ **Theorem 41.** *The languages $\mathcal{L}_{P,D,NM,F}$ and $\mathcal{L}_{P,C,NM,F}$ are equally expressive.*

All the intensional fully symmetric languages are equally expressive. Clearly the languages $\mathcal{L}_{M,-,I,F}$ and $\mathcal{L}_{-,D,I,F}$ can be trivially validly encoded into the languages $\mathcal{L}_{P,-,I,F}$ and $\mathcal{L}_{-,C,I,F}$, respectively. The following two results complete the equivalence between all the $\mathcal{L}_{-,-,I,F}$ languages.

▶ **Theorem 42.** *There exist encodings from $\mathcal{L}_{P,-,I,F}$ into $\mathcal{L}_{M,-,I,F}$.*

▶ **Theorem 43.** *There exist encodings from $\mathcal{L}_{-,C,I,F}$ into $\mathcal{L}_{-,D,I,F}$.*

6.4 Encodings into Polyadic Non-Intensional Symmetric Languages

This section considers encodings into polyadic non-intensional fully symmetric languages. Despite being nominally no-matching it is still possible to encode polyadic name-matching into the languages $\mathcal{L}_{P,-,NO,F}$. Beyond this the usual increases in expressiveness hold for shifting from monadic to polyadic, and from no-matching to name-matching. The rest of this section details these results, beginning with the relation to communication symmetric languages.

Fully symmetric communication exploits pattern unification that allows equivalence of patterns. The key difference is that a single name can unify with itself unlike in the poly-match rule where $\textsc{Match}(a, a)$ is undefined. This breaks the directionality assumed in asymmetric and communication symmetric primitives, and so invalidates many prior results dependent upon this.

The directionality of asymmetric or communication-symmetric languages can be maintained by an encoding when the target language is either polyadic or intensional.

Define the *unprotect* function g that replaces all instances of $\ulcorner a \urcorner$ with a in a pattern. Now consider the encoding $[\![\cdot]\!]$ from $\mathcal{L}_{P,D,NM,CS}$ to $\mathcal{L}_{P,C,NO,F}$ that exploits the functions Bin and Not of Section 5.2 and is the homeomorphic on all forms except the action and co-action defined as follows:

$$[\![(\widetilde{p}).P]\!] \stackrel{\text{def}}{=} a(\lambda \mathsf{rn}, \widetilde{g(p)}).[\![P]\!] \quad a = \textsc{Bin}(\widetilde{p}) \qquad [\![\langle \widetilde{p} \rangle.P]\!] \stackrel{\text{def}}{=} (\nu \mathsf{rn})a(\mathsf{rn}, \widetilde{g(p)}).[\![P]\!] \quad a = \textsc{Not}(\textsc{Bin}(\widetilde{p}))$$

The binary encoding is used to ensure that inputs and outputs are properly aligned since otherwise two outputs may unify. The additional reserved name is to distinguish actions from co-actions in the translation. Lastly, the unprotect function g converts name-matches into names.

▶ **Theorem 44.** *The encoding from $\mathcal{L}_{P,D,NM,CS}$ into $\mathcal{L}_{P,C,NO,F}$ is valid.*

Observe that since $\mathcal{L}_{P,D,NM,CS}$ generalises the languages $\mathcal{L}_{-,-,\gamma,\delta}$ where $\gamma \leq NM$ and $\delta \leq CS$ this proof can be applied to all such languages.

Relating to other fully symmetric languages, clearly $\mathcal{L}_{P,-,NO,F}$ are sub-languages of $\mathcal{L}_{P,-,NM,F}$ and can be validly encoded in them, with shifts between dataspace-based and channel-based communication handled by Theorems 40 & 41 and sub-language inclusion. The following proves an increase in expressiveness using the same technique as Theorem 35.

▶ **Theorem 45.** *There exists no valid encoding of $\mathcal{L}_{P,-,NM,F}$ into $\mathcal{L}_{P,-,NO,F}$.*

The relations between polyadic and monadic languages are as expected. $\mathcal{L}_{M,C,NO,F}$ is a sub-language of $\mathcal{L}_{P,C,NO,F}$ and can be encoded in it, and thus also $\mathcal{L}_{P,D,NO,F}$ by Theorem 40. The following separation result that $\mathcal{L}_{P,-,NO,F}$ is more expressive than $\mathcal{L}_{M,C,NO,F}$ is proven via matching degree.

▶ **Theorem 46.** *There exists no valid encoding of* $\mathcal{L}_{P,-,NO,F}$ *into* $\mathcal{L}_{M,C,NO,F}$.

Similar results hold for the name-matching languages also. $\mathcal{L}_{M,C,NM,F}$ is a sub-language of $\mathcal{L}_{P,C,NM,F}$ and can be validly encoded in it (and thus also $\mathcal{L}_{P,D,NM,F}$ by Theorem 41). The reverse encoding is by the following separation result that is proven via matching degree.

▶ **Theorem 47.** *There exists no valid encoding of* $\mathcal{L}_{P,-,NM,F}$ *into* $\mathcal{L}_{M,C,NM,F}$.

6.5 Intensional Fully Symmetric Languages

Since all the intensional fully symmetric languages are equivalent by Theorems 42 & 43, it remains to show their relations to other languages.

The intensional fully symmetric languages can also encode directionality in a similar manner to Section 6.4 (Theorem 44). Consider the encoding $[\![\cdot]\!]$ from $\mathcal{L}_{P,D,I,CS}$ to $\mathcal{L}_{M,C,I,F}$ that exploits the numerical encoding function h and Nor of Sections 5.2 & 6.4 and is the homeomorphic on all forms except:

$$[\![(p_1, \ldots, p_i).P]\!] \stackrel{\text{def}}{=} a(\lambda \mathsf{rn} \bullet (p_1 \bullet \ldots \bullet p_i)).[\![P]\!] \quad a = h(\widetilde{p})$$
$$[\![\langle p_1, \ldots, p_i \rangle.P]\!] \stackrel{\text{def}}{=} a(\mathsf{rn} \bullet (p_1 \bullet \ldots \bullet p_i)).[\![P]\!] \quad a = \mathrm{Nor}(h(\widetilde{p}))$$

where rn is a reserved name of the encoding as usual. The translations of actions and co-actions are as before except that compounding is used in place of sequencing of the polyadic patterns.

▶ **Theorem 48.** *The encoding from* $\mathcal{L}_{P,D,I,CS}$ *into* $\mathcal{L}_{M,C,I,F}$ *is valid.*

That $\mathcal{L}_{-,-,I,F}$ are more expressive than $\mathcal{L}_{P,-,I,CS}$ follows from the next two theorems.

▶ **Theorem 49.** *There exist valid encodings from* $\mathcal{L}_{P,-,I,CS}$ *into* $\mathcal{L}_{-,-,I,F}$.

▶ **Theorem 50.** *There exist no valid encodings from* $\mathcal{L}_{-,-,I,F}$ *into* $\mathcal{L}_{P,-,I,CS}$.

Finally, within the fully symmetric languages intensionality remains more expressive than non-intensionality. By Theorems 41 & 42 in all languages considered here being channel-based or dataspace-based is immaterial. The languages $\mathcal{L}_{P,-,NM,F}$ are sub-languages of $\mathcal{L}_{P,-,I,F}$ and so their expressiveness is included. The reverse is proved in the same manner as Theorem 27.

▶ **Theorem 51.** *There exist no valid encodings of* $\mathcal{L}_{P,-,I,F}$ *into* $\mathcal{L}_{P,-,NM,F}$.

7 Conclusions

Symmetric communication primitives provide new and interesting perspectives on how languages and communication can occur. By considering the intersections of various features and their relative expressiveness the position of existing calculi can be better understood, as well as that of new calculi.

Communication symmetry provides interesting insight into how much information exchange can be captured by asymmetric languages. The ability to encode polyadic communication-symmetry without name matching into asymmetric languages indicates that it is the addition of (unbounded, or multiple) name-matching with symmetry that really extends expressiveness. While communication symmetry generally increases expressiveness over asymmetric languages, it is pattern-matching that provides the strongest expressiveness alone.

The fully symmetric languages cannot be encoded into even communication symmetric languages. The flexibility of pattern-unification allows for names to be matched even in a language that nominally does not have name-matching. This yields some interesting results where languages without any pattern-matching can encode languages that have name-matching by exploiting symmetric communication. However, name-matching still provides increased expressive power within fully symmetric languages, and intensionality is the sole factor in determining the most expressive language.

This appendix contains figures of the results (Appendix A), the full results and proofs (Appendix B), and discussions (Appendix C) omitted from the body of the paper for space reasons.

A Appendix: Diagrams

The diagram in Figure 5 (repeat of Figure 4) shows all the relations between languages in this paper. All arrows indicate increases in expressive power. Black arrows are known from prior work, green arrows represent results from Section 5, and blue arrows those from Section 6. Note that since valid encodings can be composed to yield new valid encodings, no transitive relations are presented here.

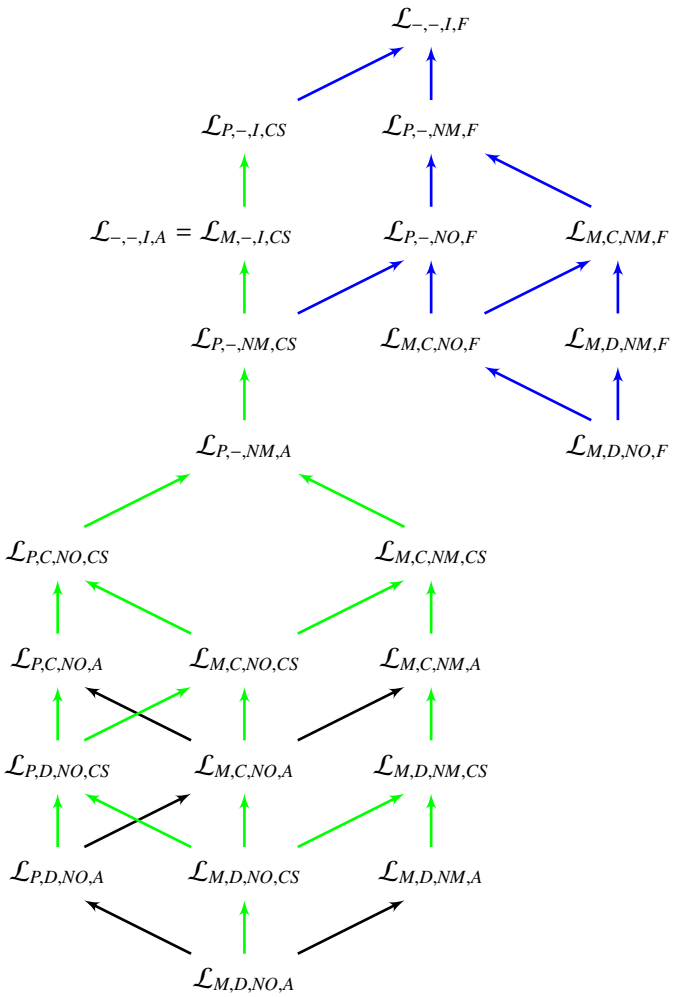

Figure 5 Relations between all languages

Figure 6 shows the relations within each language grouped according to their symmetry. Again the black arrows are prior work, green arrows are from Section 5, and blue arrows are from Section 6.

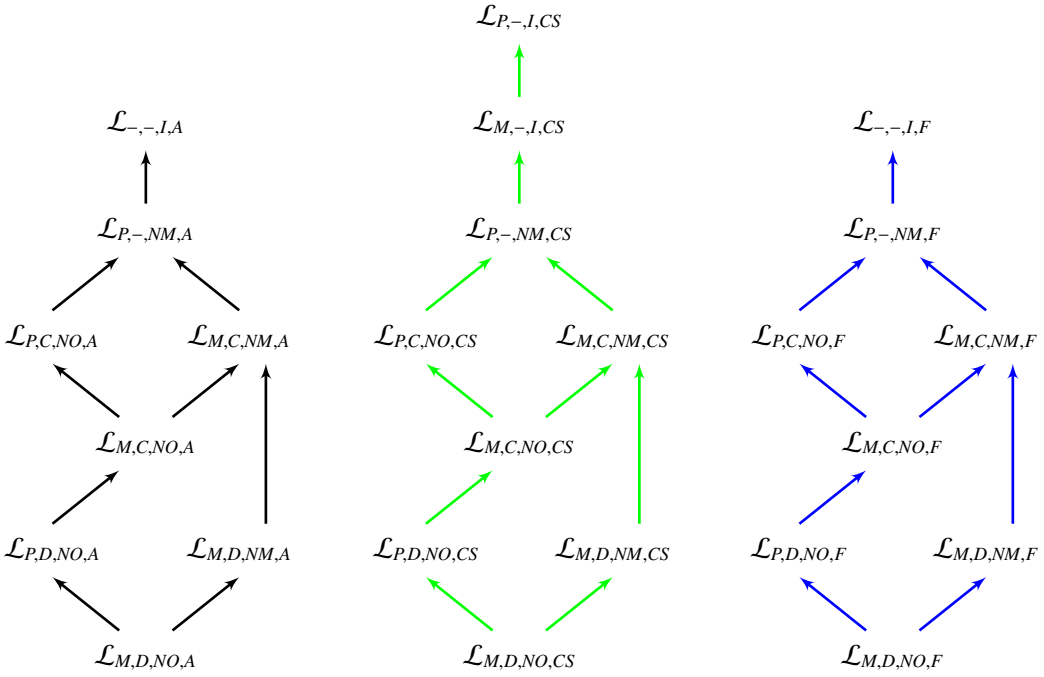

Figure 6 Relations within each language group

B Appendix: Results

The first two digits of results with numbers greater than 1000 refer to the appropriate section and subsection of the paper, e.g. Lemma 1201 fits in section 1, subsection 2.

B.1 Omitted from Section 5

▶ **Theorem 6.** *There exists no valid encoding of $\mathcal{L}_{M,D,NO,CS}$ into $\mathcal{L}_{M,D,NO,A}$.*

Proof. Assume that there exists a valid encoding $[\![\cdot]\!]$ from $\mathcal{L}_{M,D,NO,CS}$ into $\mathcal{L}_{M,D,NO,A}$. Now consider the four processes $P_1 = (\lambda x).\sqrt{}$ and $P_2 = \langle \lambda x \rangle$ and $P_3 = \langle a \rangle$ and $P_4 = (a)$. By operational correspondence none of $[\![P_i]\!]$ for $i \in \{1..4\}$ or $[\![P_1 \mid P_2]\!]$ or $[\![P_3 \mid P_4]\!]$ reduce, however since $(P_1 \mid P_2) \mid P_3$ reduces and reports success it follows that $C^N_{|}([\![P_1 \mid P_2]\!], [\![P_3]\!])$ does also. Since $[\![P_1 \mid P_2]\!] \not\mapsto$ yet $[\![P_1 \mid P_2 \mid P_3]\!] \Downarrow$ it follows that $[\![P_1 \mid P_2]\!]$ interacts via an input (or output, but the choice does not matter). Now $[\![P_3]\!]$ must interact via an output, and so must $[\![P_4]\!]$. However, it follows $[\![P_1 \mid P_2 \mid P_4]\!] \Downarrow$ which yields contradiction. ◀

▶ **Theorem 7.** *There exists no valid encoding of $\mathcal{L}_{M,D,NM,CS}$ into $\mathcal{L}_{M,D,NM,A}$.*

Proof. The same as Theorem 7. ◀

▶ **Theorem 8.** *There exists no valid encoding of $\mathcal{L}_{M,C,NO,CS}$ into $\mathcal{L}_{M,C,NO,A}$.*

Proof. The same technique as Theorem 7 can be used here. Although it may appear possible to try to separate the rôles of an action that does input and a co-action that does input via the channel name, e.g. $[\![n(\lambda x)]\!] = na(\lambda x)$ and $[\![\overline{n}\langle \lambda x \rangle]\!] = nc(\lambda x)$ where na indicates the channel name n with input on action, and nc for input on co-action, this fails the name invariance criteria. ◀

▶ **Theorem 9.** *There exists no valid encoding of $\mathcal{L}_{M,C,NM,CS}$ into $\mathcal{L}_{M,C,NM,A}$.*

Proof. The same as Theorem 9. ◀

▶ **Theorem 10.** *The languages $\mathcal{L}_{M,D,NM,CS}$ and $\mathcal{L}_{M,C,NO,CS}$ are unrelated.*

Proof. To show there is no valid encoding from $\mathcal{L}_{M,D,NM,CS}$ into $\mathcal{L}_{M,C,NO,CS}$ it suffices to consider the processes $P_1 = (\lambda x).\mathbf{0}$ and $P_2 = (\ulcorner a \urcorner).\surd$ and $P_3 = \langle a \rangle.(\ulcorner b \urcorner).\mathbf{0}$. Observe that P_3 reduces with both P_1 and P_2, reporting success with P_2. However, take $\sigma = \{b/a, a/b\}$ and now σP_3 does not reduce with P_2. Any encoding must either: allow $[\![P_2 \mid \sigma P_3]\!]$ to reduce; or not allow $[\![P_2 \mid P_3]\!]$ to reduce; both of which contradict Proposition 3. The other direction uses the technique of Theorem 4.5 of [20]. ◀

▶ **Theorem 4101.** *There exists no valid encoding of $\mathcal{L}_{M,C,NO,CS}$ into $\mathcal{L}_{M,D,NO,CS}$.*

Proof. By Theorem 6 since the matching degree of $\mathcal{L}_{M,C,NO,CS}$ is 1 and the matching degree of $\mathcal{L}_{M,D,NO,CS}$ is 0. ◀

▶ **Theorem 4102.** *There exists no valid encoding of $\mathcal{L}_{M,D,NM,CS}$ into $\mathcal{L}_{M,D,NO,CS}$.*

Proof. By Theorem 6 since the matching degree of $\mathcal{L}_{M,D,NM,CS}$ is 1 and the matching degree of $\mathcal{L}_{M,D,NO,CS}$ is 0. ◀

▶ **Theorem 4103.** *There exists no valid encoding of $\mathcal{L}_{M,C,NM,CS}$ into $\mathcal{L}_{M,D,NM,CS}$.*

Proof. By Theorem 6 since the matching degree of $\mathcal{L}_{M,C,NM,CS}$ is 2 and the matching degree of $\mathcal{L}_{M,D,NM,CS}$ is 1. ◀

▶ **Theorem 4104.** *There exists no valid encoding of $\mathcal{L}_{M,C,NM,CS}$ into $\mathcal{L}_{M,C,NO,CS}$.*

Proof. By Theorem 6 since the matching degree of $\mathcal{L}_{M,C,NM,CS}$ is 2 and the matching degree of $\mathcal{L}_{M,C,NO,CS}$ is 1. ◀

▶ **Lemma 4201.** *Given $\mathcal{L}_{M,D,NO,CS}$ action P and co-action Q then $[\![P]\!] \mid [\![Q]\!] \longmapsto$ if and only if $P \mid Q \longmapsto$.*

Proof. The proof is by definition of the poly-match rule. ◀

▶ **Lemma 4202.** *If $P \equiv Q$ then $[\![P]\!] \equiv [\![Q]\!]$. Conversely, if $[\![P]\!] \equiv Q$ then $Q = [\![P']\!]$ for some $P' \equiv P$.*

Proof. Straightforward, from the fact that $\equiv$ acts only on operators that $[\![\cdot]\!]$ translates homomorphically. ◀

▶ **Lemma 4203.** *The translation $[\![\cdot]\!]$ from $\mathcal{L}_{M,D,NO,CS}$ into $\mathcal{L}_{M,C,NO,A}$ preserves and reflects reductions. That is:*
1. *If $P \longmapsto P'$ then $[\![P]\!] \longmapsto [\![P']\!]$;*
2. *if $[\![P]\!] \longmapsto Q$ then $Q = [\![P']\!]$ for some P' such that $P \longmapsto P'$.*

Proof. Both parts can be proved by straightforward induction on the judgements $P \longmapsto P'$ and $[\![P]\!] \longmapsto Q$, respectively. In both cases, the base step is the most interesting and follows from Lemma 4201, for the second case the step $Q \longmapsto Q'$ is ensured by the definition of the translation and match rule. The inductive cases where the last rule used is a structural one then rely on Lemma 4202. ◀

▶ **Theorem 11.** *The encoding from $\mathcal{L}_{M,D,NO,CS}$ into $\mathcal{L}_{M,C,NO,A}$ is valid.*

Proof. Compositionality and name invariance hold by construction. Operational correspondence (with structural equivalence in the place of $\simeq$) and divergence reflection follow from Lemma 4203. Success sensitiveness can be proved as follows: $P \Downarrow$ means that there exists P' and $k \geq 0$ such that $P \longmapsto^k P' \equiv P'' \mid \surd$; by exploiting Lemma 4203 k times and Lemma 4202 obtain that $[\![P]\!] \longmapsto^k [\![P']\!] \equiv [\![P'']\!] \mid \surd$, i.e. that $[\![P]\!] \Downarrow$. The converse implication can be proved similarly. ◀

▶ **Theorem 12.** *There exists no valid encoding from $\mathcal{L}_{M,C,NO,A}$ into $\mathcal{L}_{M,D,NO,CS}$.*

Proof. That $\mathcal{L}_{M,C,NO,A}$ cannot be encoded by $\mathcal{L}_{M,D,NO,CS}$ follows from Theorem 6 since the matching degree of $\mathcal{L}_{M,C,NO,A}$ is 1 and the matching degree of $\mathcal{L}_{M,D,NO,CS}$ is 0. ◀

▶ **Lemma 4204.** *Given $\mathcal{L}_{M,D,NM,CS}$ action P and co-action Q then $[\![P]\!] \mid [\![Q]\!] \longmapsto$ if and only if $P \mid Q \longmapsto$.*

Proof. The proof is by definition of the poly-match rule. ◀

▶ **Lemma 4205.** *If $P \equiv Q$ then $[\![P]\!] \equiv [\![Q]\!]$. Conversely, if $[\![P]\!] \equiv Q$ then $Q = [\![P']\!]$ for some $P' \equiv P$.*

Proof. Straightforward, from the fact that $\equiv$ acts only on operators that $[\![\cdot]\!]$ translates homomorphically. ◀

▶ **Lemma 4206.** *The translation $[\![\cdot]\!]$ from $\mathcal{L}_{M,D,NM,CS}$ into $\mathcal{L}_{M,C,NM,A}$ preserves and reflects reductions. That is:*

1. *If $P \longmapsto P'$ then $[\![P]\!] \longmapsto [\![P']\!]$;*
2. *if $[\![P]\!] \longmapsto Q$ then $Q = [\![P']\!]$ for some P' such that $P \longmapsto P'$.*

Proof. Both parts can be proved by straightforward induction on the judgements $P \longmapsto P'$ and $[\![P]\!] \longmapsto Q$, respectively. In both cases, the base step is the most interesting and follows from Lemma 4204, for the second case the step $Q \longmapsto Q'$ is ensured by the definition of the translation and match rule. The inductive cases where the last rule used is a structural one then rely on Lemma 4205. ◀

▶ **Theorem 13.** *The encoding from $\mathcal{L}_{M,D,NM,CS}$ into $\mathcal{L}_{M,C,NM,A}$ is valid.*

Proof. Compositionality and name invariance hold by construction. Operational correspondence (with structural equivalence in the place of $\simeq$) and divergence reflection follow from Lemma 4206. Success sensitiveness can be proved as follows: $P \Downarrow$ means that there exists P' and $k \geq 0$ such that $P \longmapsto^k P' \equiv P'' \mid \surd$; by exploiting Lemma 4206 k times and Lemma 4205 obtain that $[\![P]\!] \longmapsto^k [\![P']\!] \equiv [\![P'']\!] \mid \surd$, i.e. that $[\![P]\!] \Downarrow$. The converse implication can be proved similarly. ◀

▶ **Theorem 14.** *There exists no valid encoding from $\mathcal{L}_{M,C,NM,A}$ into $\mathcal{L}_{M,D,NM,CS}$.*

Proof. That $\mathcal{L}_{M,C,NM,A}$ cannot be encoded by $\mathcal{L}_{M,D,NM,CS}$ follows from Theorem 6 since the matching degree of $\mathcal{L}_{M,C,NO,A}$ is 2 and the matching degree of $\mathcal{L}_{M,D,NO,CS}$ is 1. ◀

▶ **Theorem 15.** *There exist valid encodings from $\mathcal{L}_{M,C,NM,CS}$ into $\mathcal{L}_{P,-,NM,A}$.*

Proof. The proof is a straightforward adaption of the encoding and proof of Theorem 14. ◀

▶ **Theorem 16.** *There exists no valid encoding from $\mathcal{L}_{P,-,NM,A}$ into $\mathcal{L}_{M,C,NM,CS}$.*

Proof. By Theorem 6 since the matching degree of $\mathcal{L}_{P,-,NM,A}$ is ∞ and the matching degree of $\mathcal{L}_{M,C,NM,CS}$ is 2. ◀

▶ **Lemma 4207.** *Given $\mathcal{L}_{P,D,NO,CS}$ action P and co-action Q then $[\![P]\!] \mid [\![Q]\!] \longmapsto$ if and only if $P \mid Q \longmapsto$.*

Proof. The proof is by induction on the arity of the action and by definition of the poly-match rule. ◀

▶ **Lemma 4208.** *If $P \equiv Q$ then $[\![P]\!] \equiv [\![Q]\!]$. Conversely, if $[\![P]\!] \equiv Q$ then $Q = [\![P']\!]$ for some $P' \equiv P$.*

Proof. Straightforward, from the fact that $\equiv$ acts only on operators that $[\![\cdot]\!]$ translates homomorphically. ◀

▶ **Lemma 4209.** *The translation $[\![\cdot]\!]$ from $\mathcal{L}_{P,D,NO,CS}$ into $\mathcal{L}_{P,C,NO,A}$ preserves and reflects reductions. That is:*

1. *If $P \longmapsto P'$ then $[\![P]\!] \longmapsto [\![P']\!]$;*
2. *if $[\![P]\!] \longmapsto Q$ then $Q = [\![P']\!]$ for some P' such that $P \longmapsto P'$.*

Proof. Both parts can be proved by straightforward induction on the judgements $P \longmapsto P'$ and $[\![P]\!] \longmapsto Q$, respectively. In both cases, the base step is the most interesting and follows from Lemma 4207, for the second case the step $Q \longmapsto Q'$ is ensured by the definition of the translation and match rule. The inductive cases where the last rule used is a structural one then rely on Lemma 4208. ◀

▶ **Theorem 17.** *The encoding from $\mathcal{L}_{P,D,NO,CS}$ into $\mathcal{L}_{P,C,NO,A}$ is valid.*

Proof. Compositionality and name invariance hold by construction. Operational correspondence (with structural equivalence in the place of $\simeq$) and divergence reflection follow from Lemma 4209. Success sensitiveness can be proved as follows: $P \Downarrow$ means that there exists P' and $k \geq 0$ such that $P \longmapsto^k P' \equiv P'' \mid \surd$; by exploiting Lemma 4209 k times and Lemma 4208 obtain that $[\![P]\!] \longmapsto^k [\![P']\!] \equiv [\![P'']\!] \mid \surd$, i.e. that $[\![P]\!] \Downarrow$. The converse implication can be proved similarly. ◀

▶ **Theorem 18.** *There exists no valid encoding from $\mathcal{L}_{P,C,NO,A}$ into $\mathcal{L}_{P,D,NO,CS}$.*

Proof. By Theorem 6 since the matching degree of $\mathcal{L}_{P,C,NO,A}$ is 1 and the matching degree of $\mathcal{L}_{P,D,NO,CS}$ is 0. ◀

▶ **Theorem 19.** *There exists no valid encoding of $\mathcal{L}_{P,D,NO,CS}$ into $\mathcal{L}_{M,D,NO,CS}$.*

Proof. The proof is by contradiction. Assume there exists a valid encoding $[\![\cdot]\!]$ from $\mathcal{L}_{P,D,NO,CS}$ into $\mathcal{L}_{M,D,NO,CS}$. Consider the processes $P_1 = (\lambda x, b).\mathbf{if}\ x = a\ \mathbf{then}\ (\lambda x).\surd\ \mathbf{else}$ and $P_2 = (\lambda y, d).\mathbf{0}$ and $P_3 = \langle c, \lambda x\rangle.\mathbf{if}\ x = b\ \mathbf{then}\ \langle b\rangle.\surd\ \mathbf{else}$ and $P_4 = \langle a, \lambda y\rangle.\mathbf{0}$. Since $P_1 \mid P_2 \mid P_3 \mid P_4 \not\Downarrow$ it follows that $[\![\, P_1 \mid P_2 \mid P_3 \mid P_4 \,]\!] \not\Downarrow$. However, it can be shown that either $[\![\, P_1 \mid P_2 \mid P_3 \mid P_4 \,]\!] \not\longmapsto$ or $[\![\, P_1 \mid P_2 \mid P_3 \mid P_4 \,]\!] \Downarrow$, both of which yield contradiction. ◀

▶ **Theorem 20.** *There exists no valid encoding of $\mathcal{L}_{P,C,NO,CS}$ into $\mathcal{L}_{M,C,NO,CS}$.*

Proof. The same technique as Theorem 20 can be used here by having all communication along the same channel name and using name invariance to prevent modification of the channel name by the encoding. ◀

▶ **Theorem 21.** *There exists no valid encoding of $\mathcal{L}_{P,D,NO,CS}$ into $\mathcal{L}_{P,D,NO,A}$.*

Proof. The proof is by contradiction. Assume there is a valid encoding of $\mathcal{L}_{P,D,NO,CS}$ into $\mathcal{L}_{P,D,NO,A}$. The key to the proof is to consider the processes $P_1 = \textbf{if } x = a \textbf{ then } (\lambda z).\surd$ and $P_2 = \textbf{if } y = d \textbf{ then } \langle z \rangle.\mathbf{0}$. Clearly neither $(\lambda x, b).P_1 \mid \langle a, \lambda y \rangle.P_2$ nor $(\lambda x, d).\mathbf{0} \mid \langle c, \lambda y \rangle.\mathbf{0}$ report success and by validity of the encoding their encodings do not either. Conclude by showing that the process $((\lambda x, b).P_1 \mid \langle a, \lambda y \rangle.P_2) \mid ((\lambda x, d).\mathbf{0} \mid \langle c, \lambda y \rangle.\mathbf{0})$ does not report success while showing that its encoding $[\![((\lambda x, b).P_1 \mid \langle a, \lambda y \rangle.P_2) \mid ((\lambda x, d).\mathbf{0} \mid \langle c, \lambda y \rangle.\mathbf{0})]\!]$ does. ◂

▸ **Lemma 4401.** *Given $\mathcal{L}_{M,C,I,CS}$ action P and co-action Q then $[\![P]\!] \mid [\![Q]\!] \longmapsto$ if and only if $P \mid Q \longmapsto$.*

Proof. The proof is by induction on the arity of the action and by definition of the poly-match rule. ◂

▸ **Lemma 4402.** *If $P \equiv Q$ then $[\![P]\!] \equiv [\![Q]\!]$. Conversely, if $[\![P]\!] \equiv Q$ then $Q = [\![P']\!]$ for some $P' \equiv P$.*

Proof. Straightforward, from the fact that $\equiv$ acts only on operators that $[\![\cdot]\!]$ translates homomorphically. ◂

▸ **Lemma 4403.** *The translation $[\![\cdot]\!]$ from $\mathcal{L}_{M,C,I,CS}$ into $\mathcal{L}_{M,D,I,CS}$ preserves and reflects reductions. That is:*
1. *If $P \longmapsto P'$ then $[\![P]\!] \longmapsto [\![P']\!]$;*
2. *if $[\![P]\!] \longmapsto Q$ then $Q = [\![P']\!]$ for some P' such that $P \longmapsto P'$.*

Proof. Both parts can be proved by straightforward induction on the judgements $P \longmapsto P'$ and $[\![P]\!] \longmapsto Q$, respectively. In both cases, the base step is the most interesting and follows from Lemma 4401, for the second case the step $Q \longmapsto Q'$ is ensured by the definition of the translation and match rule. The inductive cases where the last rule used is a structural one then rely on Lemma 4402. ◂

▸ **Theorem 22.** *The encoding from $\mathcal{L}_{M,C,I,CS}$ into $\mathcal{L}_{M,D,I,CS}$ is valid.*

Proof. Compositionality and name invariance hold by construction. Operational correspondence (with structural equivalence in the place of $\simeq$) and divergence reflection follow from Lemma 4403. Success sensitiveness can be proved as follows: $P \Downarrow$ means that there exists P' and $k \geq 0$ such that $P \longmapsto^k P' \equiv P'' \mid \surd$; by exploiting Lemma 4403 k times and Lemma 4402 obtain that $[\![P]\!] \longmapsto^k [\![P']\!] \equiv [\![P'']\!] \mid \surd$, i.e. that $[\![P]\!] \Downarrow$. The converse implication can be proved similarly. ◂

▸ **Theorem 23.** *There is a valid encoding from $\mathcal{L}_{M,D,I,CS}$ into $\mathcal{L}_{M,C,I,A}$.*

Proof. The proof is a straightforward adaption of the encoding and proof of Theorem 12. ◂

▸ **Lemma 4404.** *Given $\mathcal{L}_{P,C,NM,CS}$ action P and co-action Q then $[\![P]\!] \mid [\![Q]\!] \longmapsto$ if and only if $P \mid Q \longmapsto$.*

Proof. The proof is by induction on the arity of the interaction and by definition of the poly-match rule. ◂

▸ **Lemma 4405.** *If $P \equiv Q$ then $[\![P]\!] \equiv [\![Q]\!]$. Conversely, if $[\![P]\!] \equiv Q$ then $Q = [\![P']\!]$ for some $P' \equiv P$.*

Proof. Straightforward, from the fact that $\equiv$ acts only on operators that $[\![\cdot]\!]$ translates homomorphically. ◂

▸ **Lemma 4406.** *The translation $[\![\cdot]\!]$ from $\mathcal{L}_{P,C,NM,CS}$ into $\mathcal{L}_{P,D,NM,CS}$ preserves and reflects reductions. That is:*
1. *If $P \longmapsto P'$ then $[\![P]\!] \longmapsto [\![P']\!]$;*

2. if $[\![P]\!] \longmapsto Q$ *then* $Q = [\![P']\!]$ *for some* P' *such that* $P \longmapsto P'$.

Proof. Both parts can be proved by straightforward induction on the judgements $P \longmapsto P'$ and $[\![P]\!] \longmapsto Q$, respectively. In both cases, the base step is the most interesting and follows from Lemma 4404, for the second case the step $Q \longmapsto Q'$ is ensured by the definition of the translation and match rule. The inductive cases where the last rule used is a structural one then rely on Lemma 4405. ◀

▶ **Theorem 24.** *The encoding from* $\mathcal{L}_{P,C,NM,CS}$ *into* $\mathcal{L}_{P,D,NM,CS}$ *is valid.*

Proof. Compositionality and name invariance hold by construction. Operational correspondence (with structural equivalence in the place of $\simeq$) and divergence reflection follow from Lemma 4406. Success sensitiveness can be proved as follows: $P \Downarrow$ means that there exists P' and $k \geq 0$ such that $P \longmapsto^k P' \equiv P'' \mid \surd$; by exploiting Lemma 4406 k times and Lemma 4405 obtain that $[\![P]\!] \longmapsto^k [\![P']\!] \equiv [\![P'']\!] \mid \surd$, i.e. that $[\![P]\!] \Downarrow$. The converse implication can be proved similarly. ◀

▶ **Theorem 25.** *The exists a valid encoding from* $\mathcal{L}_{P,C,I,CS}$ *into* $\mathcal{L}_{P,D,I,CS}$.

Proof. In the same manner as Theorem 25. ◀

▶ **Theorem 26.** *There exists no valid encoding of* $\mathcal{L}_{P,-,I,CS}$ *into* $\mathcal{L}_{P,-,NM,CS}$.

Proof. The proof technique is the same as Theorem 7.1 of [15]. The technique exploits the arity k of the reduction between the encoded processes $P_0 = (\lambda x).\langle m \rangle.\mathbf{0}$ and $P_1 = \langle a \rangle.\mathbf{0}$ to show that another encoded process $P_2 = \langle a_1 \bullet \ldots \bullet a_{k+2} \rangle.\mathbf{0}$ must either: interact with arity $< k+2$ (and then fail to match some name a_i and contradict operational correspondence); or result in divergent computation. ◀

▶ **Theorem 27.** *There exists no valid encoding from* $\mathcal{L}_{M,-,I,CS}$ *into* $\mathcal{L}_{P,-,NM,CS}$.

Proof. The proof of Theorem 27 can be used here. ◀

▶ **Theorem 28.** *There exists no valid encoding from* $\mathcal{L}_{P,-,NM,CS}$ *into* $\mathcal{L}_{M,-,I,CS}$.

Proof. The proof technique of Theorem 20 can be used here. ◀

▶ **Theorem 29.** *There exists no valid encoding from* $\mathcal{L}_{P,-,NM,CS}$ *into* $\mathcal{L}_{P,-,NM,A}$.

Proof. The proof is by contradiction, assume there exists a valid encoding $[\![\cdot]\!]$. Consider the following processes $P_1 = (a, \ulcorner b \urcorner).(\ulcorner m \urcorner).\surd$ and $P_2 = \langle \ulcorner c \urcorner, d \rangle.\langle m \rangle.\mathbf{0}$ where a and b and c and d are pairwise distinct names. Consider the substitutions $\sigma_1 = \{a/c\}$ and $\sigma_2 = \{b/d\}$. Observe that none of $P_1 \mid P_2$ or $\sigma_1(P_1 \mid P_2)$ or $\sigma_2(P_1 \mid P_2)$ reduce or report success, and thus their encodings do not either. However, $\sigma_1(\sigma_2(P_1 \mid P_2))$ does reduce and report success, and thus there exists σ'_1 and σ'_2 such that $R = \sigma'_1(\sigma'_2([\![P_1 \mid P_2]\!]))$ does reduce and report success. Now consider the names matched in the first reduction of $R \longmapsto R'$. Clearly if the names are not in the union of the ranges of σ'_1 and σ'_2 then $\sigma'_1([\![P_1 \mid P_2]\!])$ or $\sigma'_2([\![P_1 \mid P_2]\!])$ or $[\![P_1 \mid P_2]\!]$ would reduce, but this would contradict a validity of the encoding. Now by exploiting the process $P_3 = \langle \ulcorner c \urcorner, \lambda z \rangle.\langle m, d \rangle.\mathbf{0}$ it can be shown that either $\sigma'_1(\sigma'_2([\![P_1 \mid P_3]\!])) \mapsto\!\!\!\!\!\!\not\;\;$ or $\sigma'_1(\sigma'_2([\![P_1 \mid P_2 \mid P_3]\!]))$ is divergent, both of which yield contradiction. ◀

▶ **Theorem 30.** *There exists no valid encoding from* $\mathcal{L}_{P,-,I,CS}$ *into* $\mathcal{L}_{M,-,I,CS}$.

Proof. The proof technique as Theorem 30 can be used here. ◀

B.2 Omitted from Section 6

▶ **Lemma 5101.** *Given a language $\mathcal{L}_1$ that it not fully symmetric, then for all $\mathcal{L}_1$ processes S such that $S \mapsto\!\!\!\!\!\!\rightarrow$, it holds that $S \mid S \mapsto\!\!\!\!\!\!\rightarrow$.*

Proof. By contradiction. Assume that $S \mid S \longmapsto$ and consider how this reduction occurred. If the reduction is due to $S \longmapsto S'$ then this contradicts $S \mapsto\!\!\!\!\!\!\rightarrow$.

It follows that the reduction must be of some $a(\widetilde{p}).S_1 \mid \overline{a}\langle\widetilde{q}\rangle.S_2 \longmapsto \sigma S_1 \mid \rho S_2$ where $\textsc{Match}(\widetilde{p};\widetilde{q}) = (\sigma,\rho)$ and $S \mid S \equiv (\nu\widetilde{n})(a(\widetilde{p}).S_1 \mid \overline{a}\langle\widetilde{q}\rangle.S_2 \mid R)$ for some R (the a's are omitted in the dataspace-based languages). It is straightforward (if tedious) to show that both $a(\widetilde{p}).S_1$ and $\overline{a}\langle\widetilde{q}\rangle.S_2$ must be contained within S, and that for the reduction to occur it must be possible for $S \longmapsto$ yielding contradiction. ◀

▶ **Theorem 31.** *There exists no valid encoding of a fully symmetric language $\mathcal{L}_{-,-,-,F}$ into any asymmetric or communication symmetric language $\mathcal{L}_{-,-,-,\delta}$ $\delta \neq F$.*

Proof. Observe that in all the fully symmetric languages the self recognising process S is such that $S \mapsto\!\!\!\!\!\!\rightarrow$ and $S \Downarrow\!\!\!\!\!\!/$, however $S \mid S \longmapsto$ and $S \mid S \Downarrow$. Conclude by Theorem 4 and Lemma 5101. ◀

▶ **Theorem 5201.** *There exists no valid encoding from $\mathcal{L}_{M,D,NO,F}$ into $\mathcal{L}_{M,D,NO,\delta}$ where $\delta \neq F$.*

Proof. By Theorem 32. ◀

▶ **Theorem 5202.** *There exists no valid encoding from $\mathcal{L}_{M,\beta,\gamma,F}$ into $\mathcal{L}_{M,\beta,\gamma,\delta}$ where $\gamma \leq NM$ and $\delta \neq F$.*

Proof. By Theorem 32. ◀

▶ **Theorem 32.** *There exists no valid encoding from $\mathcal{L}_{M,D,NO,\delta}$ where $\delta \neq F$ into $\mathcal{L}_{M,D,NO,F}$.*

Proof. By contradiction. Assume there exists a valid encoding $[\![\cdot]\!]$ from $\mathcal{L}_{M,D,NO,\delta}$ where $\delta \neq F$ into $\mathcal{L}_{M,D,NO,F}$. Consider the $\mathcal{L}_{M,D,NO,\delta}$ processes $P_1 = (\lambda x).\surd$ and $P_2 = \langle a\rangle.\surd$. Since neither of P_1 or P_2 reduce or report success it follows that their encodings do not. Since $P_1 \mid P_2$ does reduce and report success it follows that $[\![P_1 \mid P_2]\!]$ also reduces and reports success. By considering the context $C^N_{|}([\![P_1]\!], [\![P_2]\!])$ the reduction must be due to some $(p).S'$ and some $(b).T'$ for some p and b and S' and T'. By validity of the encoding it must be that $(b).T'$ is part of $[\![P_i]\!]$ for $i \in \{1,2\}$. Observe that by definition of the poly-unify rule $(b).T' \mid (b).T'$ reduces. Now the context $C^N_{|}([\![P_i]\!], [\![P_i]\!])$ can be shown to reduce (and report success), however this contradicts that $P_i \mid P_i \mapsto\!\!\!\!\!\!\rightarrow$. ◀

▶ **Theorem 33.** *There exists no valid encoding from $\mathcal{L}_{M,-,\gamma,\delta}$ into $\mathcal{L}_{M,-,\gamma,F}$ where $\gamma \leq NM$ and $\delta \neq F$.*

Proof. The same technique as Theorem 33 can be used here. ◀

▶ **Theorem 34.** *There exists no valid encoding of $\mathcal{L}_{M,D,NM,F}$ into $\mathcal{L}_{M,D,NO,F}$.*

Proof. The proof is by contradiction, assume that there exists a valid encoding $[\![\cdot]\!]$ from $\mathcal{L}_{M,D,NM,F}$ into $\mathcal{L}_{M,D,NO,F}$. Now consider the process $P_1 = (\ulcorner a \urcorner).\surd$. Clearly P_1 does not reduce or report success and so it's encoding does not either. However, $P_1 \mid P_1$ does reduce and report success, and so $[\![P_1 \mid P_1]\!]$ must also. By definition of the poly-unify rule the reduction $[\![P_1 \mid P_1]\!]$ must be between an action $(b).S'$ and another action, now consider this other action:

- If the action is $(\lambda z).S''$ then it must arise from the encoding $[\![P_1]\!]$. It follows via reasoning over the structure of $[\![P_1]\!]$ that $[\![P_1]\!]$ must be able to reduce, but this contradicts the validity of the encoding.

- If the action is $(b).S''$ then it can be shown that b must be determined by a and the encoding (otherwise $P_2 = (c).\mathbf{0}$ with $\{a, b, c\}$ pairwise distinct would yield that $[\![P_1 \mid P_2]\!]$ reduces while $P_1 \mid P_2$ does not). Now consider the processes $P_3 = (\lambda z).\mathbf{if}\ a = z\ \mathbf{then}\ \surd$ and $P_4 = (a).\mathbf{0}$. Since $P_1 \mid P_4$ reduces and reports success it follows that the encoding $[\![P_1 \mid P_4]\!]$ must do also.
  - Now if $[\![P_4]\!]$ interacts via only $(b).T$ then since $P_3 \mid P_4$ reduce then $[\![P_3]\!]$ must be able to interact with $(b).T$. However, then $[\![P_1 \mid P_3]\!]$ would reduce, which contradicts the validity of encoding.
  - If $[\![P_4]\!]$ interacts via only the form $(\lambda q).T$ then it follows that $[\![P_4 \mid P_4]\!]$ does not reduce while $P_4 \mid P_4$ does, yielding contradiction.
  - Therefore $[\![P_4]\!]$ must have multiple forms of interaction, which can in turn be used to show that the encoding is contradictory via either operational correspondence or divergence reflection.

◀

▶ **Theorem 35.** *There exists no valid encoding of $\mathcal{L}_{M,C,NM,F}$ into $\mathcal{L}_{M,C,NO,F}$.*

Proof. The same technique as Theorem 35 works here. ◀

▶ **Theorem 36.** *There exists no valid encoding of $\mathcal{L}_{M,C,NO,F}$ into $\mathcal{L}_{M,D,NO,F}$.*

Proof. Observe that the matching degree of $\mathcal{L}_{M,C,NO,F}$ is 2 and the matching degree of $\mathcal{L}_{M,D,NO,F}$ is 1, conclude by Theorem 6. ◀

▶ **Theorem 37.** *There exists no valid encoding of $\mathcal{L}_{M,C,NM,F}$ into $\mathcal{L}_{M,D,NM,F}$.*

Proof. Observe that the matching degree of $\mathcal{L}_{M,C,NM,F}$ is 2 and the matching degree of $\mathcal{L}_{M,D,NM,F}$ is 1, conclude by Theorem 6. ◀

▶ **Theorem 38.** *The languages $\mathcal{L}_{M,D,NM,F}$ and $\mathcal{L}_{M,C,NO,F}$ are unrelated.*

Proof. That there exists no valid encoding from $\mathcal{L}_{M,D,NM,F}$ into $\mathcal{L}_{M,C,NO,F}$ follows from Theorem 35. In the reverse direction the matching degree suffices to show separation. ◀

▶ **Lemma 5301.** *Given $\mathcal{L}_{P,C,NO,F}$ actions P and Q then $[\![P]\!] \mid [\![Q]\!] \longmapsto$ if and only if $P \mid Q \longmapsto$.*

Proof. The proof is by induction on the arity of P and by definition of the poly-unify rule. ◀

▶ **Lemma 5302.** *If $P \equiv Q$ then $[\![P]\!] \equiv [\![Q]\!]$. Conversely, if $[\![P]\!] \equiv Q$ then $Q = [\![P']\!]$ for some $P' \equiv P$.*

Proof. Straightforward, from the fact that $\equiv$ acts only on operators that $[\![\cdot]\!]$ translates homomorphically. ◀

▶ **Lemma 5303.** *The translation $[\![\cdot]\!]$ from $\mathcal{L}_{P,C,NO,F}$ into $\mathcal{L}_{P,D,NO,F}$ preserves and reflects reductions. That is:*

1. *If $P \longmapsto P'$ then $[\![P]\!] \longmapsto [\![P']\!]$;*
2. *if $[\![P]\!] \longmapsto Q$ then $Q = [\![P']\!]$ for some P' such that $P \longmapsto P'$.*

Proof. Both parts can be proved by straightforward induction on the judgements $P \longmapsto P'$ and $[\![P]\!] \longmapsto Q$, respectively. In both cases, the base step is the most interesting and follows from Lemma 5301, for the second case the step $Q \longmapsto Q'$ is ensured by the definition of the translation and match rule. The inductive cases where the last rule used is a structural one then rely on Lemma 5302. ◀

▶ **Theorem 39.** *The encoding from $\mathcal{L}_{P,C,NO,F}$ into $\mathcal{L}_{P,D,NO,F}$ is valid.*

Proof. Compositionality and name invariance hold by construction. Operational correspondence (with structural equivalence in the place of $\simeq$) and divergence reflection follow from Lemma 5303. Success sensitiveness can be proved as follows: $P \Downarrow$ means that there exists P' and $k \geq 0$ such that $P \longmapsto^k P' \equiv P'' \mid \surd$; by exploiting Lemma 5303 k times and Lemma 5302 obtain that $[\![P]\!] \longmapsto^k [\![P']\!] \equiv [\![P'']\!] \mid \surd$, i.e. that $[\![P]\!] \Downarrow$. The converse implication can be proved similarly. ◂

▸ **Theorem 40.** *The languages $\mathcal{L}_{P,D,NM,F}$ and $\mathcal{L}_{P,C,NM,F}$ are equally expressive.*

Proof. That $\mathcal{L}_{P,D,NM,F}$ can be validly encoded into $\mathcal{L}_{P,C,NM,F}$ is trivial. The reverse direction can be shown in the same manner as for Theorem 40. ◂

▸ **Theorem 41.** *There exist encodings from $\mathcal{L}_{P,-,I,F}$ into $\mathcal{L}_{M,-,I,F}$.*

Proof. The same technique as Theorem 5.4 of [15] can be used by encoding the polyadic structure into a monadic intensional pattern. The key idea is that a sequence of patterns $\widetilde{p} = p_1, \ldots, p_i$ is encoded as a single pattern $(\mathsf{rn} \bullet p_1) \bullet \ldots \bullet p_i$ exploiting a reserved name rn. ◂

▸ **Theorem 42.** *There exist encodings from $\mathcal{L}_{-,C,I,F}$ into $\mathcal{L}_{-,D,I,F}$.*

Proof. The same technique as used in Theorem 23 can be used here. ◂

▸ **Lemma 5401.** *Given $\mathcal{L}_{P,D,NO,CS}$ action P and co-action Q then $[\![P]\!] \mid [\![Q]\!] \longmapsto$ if and only if $P \mid Q \longmapsto$.*

Proof. The proof is by induction on the arity of the interaction and by definition of the poly-unify rule. ◂

▸ **Lemma 5402.** *If $P \equiv Q$ then $[\![P]\!] \equiv [\![Q]\!]$. Conversely, if $[\![P]\!] \equiv Q$ then $Q = [\![P']\!]$ for some $P' \equiv P$.*

Proof. Straightforward, from the fact that $\equiv$ acts only on operators that $[\![\cdot]\!]$ translates homomorphically. ◂

▸ **Lemma 5403.** *The translation $[\![\cdot]\!]$ from $\mathcal{L}_{P,D,NM,CS}$ into $\mathcal{L}_{P,C,NO,F}$ preserves and reflects reductions. That is:*

1. *If $P \longmapsto P'$ then $[\![P]\!] \longmapsto [\![P']\!]$;*
2. *if $[\![P]\!] \longmapsto Q$ then $Q = [\![P']\!]$ for some P' such that $P \longmapsto P'$.*

Proof. Both parts can be proved by straightforward induction on the judgements $P \longmapsto P'$ and $[\![P]\!] \longmapsto Q$, respectively. In both cases, the base step is the most interesting and follows from Lemma 5401, for the second case the step $Q \longmapsto Q'$ is ensured by the definition of the translation and match rule. The inductive cases where the last rule used is a structural one then rely on Lemma 5402. ◂

▸ **Theorem 43.** *The encoding from $\mathcal{L}_{P,D,NM,CS}$ into $\mathcal{L}_{P,C,NO,F}$ is valid.*

Proof. Compositionality and name invariance hold by construction. Operational correspondence (with structural equivalence in the place of $\simeq$) and divergence reflection follow from Lemma 5403. Success sensitiveness can be proved as follows: $P \Downarrow$ means that there exists P' and $k \geq 0$ such that $P \longmapsto^k P' \equiv P'' \mid \surd$; by exploiting Lemma 5403 k times and Lemma 5402 obtain that $[\![P]\!] \longmapsto^k [\![P']\!] \equiv [\![P'']\!] \mid \surd$, i.e. that $[\![P]\!] \Downarrow$. The converse implication can be proved similarly. ◂

▸ **Theorem 44.** *There exists no valid encoding of $\mathcal{L}_{P,-,NM,F}$ into $\mathcal{L}_{P,-,NO,F}$.*

Proof. The same technique as Theorem 35 can be used here to show no encoding from a name-matching fully symmetric language into a non-matching fully symmetric language. ◂

▶ **Theorem 45.** *There exists no valid encoding of $\mathcal{L}_{P,-,NO,F}$ into $\mathcal{L}_{M,C,NO,F}$.*

Proof. Observe that the matching degree of $\mathcal{L}_{P,-,NO,F}$ is ∞ and the matching degree of $\mathcal{L}_{M,C,NO,F}$ is 2, conclude by Theorem 6. ◀

▶ **Theorem 46.** *There exists no valid encoding of $\mathcal{L}_{P,-,NM,F}$ into $\mathcal{L}_{M,C,NM,F}$.*

Proof. Observe that the matching degree of $\mathcal{L}_{P,-,NM,F}$ is ∞ and the matching degree of $\mathcal{L}_{M,C,NM,F}$ is 2, conclude by Theorem 6. ◀

▶ **Lemma 5501.** *Given $\mathcal{L}_{P,D,I,CS}$ action P and co-action Q then $[\![P]\!] \mid [\![Q]\!] \longmapsto$ if and only if $P \mid Q \longmapsto$.*

Proof. The proof is by induction on the arity of P and by definition of the poly-unify rule. ◀

▶ **Lemma 5502.** *If $P \equiv Q$ then $[\![P]\!] \equiv [\![Q]\!]$. Conversely, if $[\![P]\!] \equiv Q$ then $Q = [\![P']\!]$ for some $P' \equiv P$.*

Proof. Straightforward, from the fact that $\equiv$ acts only on operators that $[\![\cdot]\!]$ translates homomorphically. ◀

▶ **Lemma 5503.** *The translation $[\![\cdot]\!]$ from $\mathcal{L}_{P,D,I,CS}$ into $\mathcal{L}_{M,C,I,F}$ preserves and reflects reductions. That is:*
1. *If $P \longmapsto P'$ then $[\![P]\!] \longmapsto [\![P']\!]$;*
2. *if $[\![P]\!] \longmapsto Q$ then $Q = [\![P']\!]$ for some P' such that $P \longmapsto P'$.*

Proof. Both parts can be proved by straightforward induction on the judgements $P \longmapsto P'$ and $[\![P]\!] \longmapsto Q$, respectively. In both cases, the base step is the most interesting and follows from Lemma 5501, for the second case the step $Q \longmapsto Q'$ is ensured by the definition of the translation and match rule. The inductive cases where the last rule used is a structural one then rely on Lemma 5502. ◀

▶ **Theorem 47.** *The encoding from $\mathcal{L}_{P,D,I,CS}$ into $\mathcal{L}_{M,C,I,F}$ is valid.*

Proof. Compositionality and name invariance hold by construction. Operational correspondence (with structural equivalence in the place of $\simeq$) and divergence reflection follow from Lemma 5503. Success sensitiveness can be proved as follows: $P \Downarrow$ means that there exists P' and $k \geq 0$ such that $P \longmapsto^k P' \equiv P'' \mid \surd$; by exploiting Lemma 5503 k times and Lemma 5502 obtain that $[\![P]\!] \longmapsto^k [\![P']\!] \equiv [\![P'']\!] \mid \surd$, i.e. that $[\![P]\!] \Downarrow$. The converse implication can be proved similarly. ◀

▶ **Theorem 48.** *There exist valid encodings from $\mathcal{L}_{P,-,I,CS}$ into $\mathcal{L}_{-,-,I,F}$.*

Proof. There exists an encoding from $\mathcal{L}_{P,D,I,CS}$ into $\mathcal{L}_{M,C,I,F}$ by Theorem 48. The rest follow by Theorems 26, 42 & 43 and by transitivity of valid encodings. ◀

▶ **Theorem 49.** *There exist no valid encodings from $\mathcal{L}_{-,-,I,F}$ into $\mathcal{L}_{P,-,I,CS}$.*

Proof. For all combinations the proof follows from Theorem 32. ◀

▶ **Theorem 50.** *There exist no valid encodings of $\mathcal{L}_{P,-,I,F}$ into $\mathcal{L}_{P,-,NM,F}$.*

Proof. The same technique as Theorem 27 applies here. ◀

C Appendix: Discussions

This section presents various points of discussion that have been omitted from the paper due to space constraints. They are presented as a guide rather than an exhaustive discussion of the topic.

Along that line, this paper does not include a significant discussion of the implication of the results here beyond short comments and illustration through the results themselves. This choice is due to three factors: First, the lack of space to properly engage with such a discussion. The work here is already significantly pared down to meet the page limits, and omitting key technical points may render the paper unreadable, while only discussing results without the technical detail may be unconvincing. Second, the scope of such a discussion would be quite significant. Even limiting to the languages presented here, this would need to account for the subtle interplay of four features and thirty-six languages. Relating them further to other languages and features while doing proper service to prior work and the broader impact is a further paper in itself. Third, it is not clear that the features here are yet complete. There are other aspects of languages that have yet to be fully explored in this framework (see Future Work below) that are due the same treatment. In that sense it may be premature to go into detail before the features and languages have been well mapped out.

C.1 Choices of Primitives

The choices of primitives for the languages here is to align with prior work [20, 22, 15]. However, it is worth noting that there are other choices that would impact some results, here and previously.

The patterns are chosen here to match those of CPC and so exploit results for CPC that have been generalised to other languages [13, 16]. It turns out that the (symmetric) patterns here are sufficient to represent most other approaches, such as Spi Calculus [18] and Psi Calculi terms [15]. More generally the core approach of compounding proves sufficient to represent many complex data structures and even (in practice) type information. This has been discussed, formalised, and reasoned about in different settings in [25, 26, 13] and in many works related to *pattern calculus* and CPC.

For the process forms the most obvious alternative would be to consider some form of the choice operator: $ACT.P + ACT.P$ or $COACT.P + COACT.P$ or $ACT.P + COACT.P$ or some variation thereof. Again the choice not to include this is to match with prior results [20, 22, 15]. The addition of such a choice operator could invalidate some findings, in particular Theorem 5.8 from [22] and Lemma 5101 (that is key to Theorem 32). This provides illustration of which results would need to be reexamined with such a change, although it does not (a priori) indicate that the overall relative expressiveness would change. For example, previous simple results for the inability to encode CPC ($\mathcal{L}_{M,D,I,F}$) into π-calculus ($\mathcal{L}_{M,C,NO,A}$) have used this approach [18], however alternative proofs also exist such as Theorems 6, 30, & 35 and in prior works [20, 15]. This lends weight to the detail here that provides alternative approaches, and identifies which results rely on which primitives in the language.

In this context there are many other possible choices of primitives for both the patterns and the processes. However, those here are sufficient to understand the core dynamics between the interaction features of languages. Also by using a common approach (valid encodings) that is transitive, often more distant relations can be proved without relying on particular choices of primitives or proof techniques.

C.2 Related Work

This section provides a brief account of related works most close to the decisions and results here, since to cover all related works would take an entire paper.

Expressiveness in process calculi and similar languages has been widely explored, even when focusing mostly upon the choice of communication primitives [34, 5, 6, 9, 24, 20, 17, 13, 18]. The

choice of valid encodings here is that used, sometimes with mild adaptations, in [22, 21, 17, 33, 13, 18] and has also inspired similar works [27, 28, 40]. However, there are alternative approaches to encoding criteria or comparing expressive power [4, 10, 6, 36, 40]. Further arguments for, and against, the valid encodings here can be found in [22, 21, 40, 18].

Many of the languages here correspond to published languages. Observe that $\mathcal{L}_{M,C,NO,A}$, and $\mathcal{L}_{P,C,NO,A}$ align with the communication primitives of the monadic/polyadic π-calculus [30, 31, 29]. The language $\mathcal{L}_{P,D,NM,A}$ aligns with LINDA[12]; the languages $\mathcal{L}_{M,D,NO,A}$ and $\mathcal{L}_{P,D,NO,A}$ with the monadic/polyadic Mobile Ambients [7]; and $\mathcal{L}_{P,C,NO,A}$ with that of μKLAIM [32] or semantic-π [8]. The asymmetric intensional languages do not usually exactly match well-known calculi. However, the language $\mathcal{L}_{M,D,I,A}$ corresponds to Asymmetric Concurrent Pattern Calculus [14] and also $\mathcal{L}_{M,C,I,A}$ has been mentioned in [13], as a variation of CPC, also both have a behavioural theory as a specialisation of [16]. Similarly, the language $\mathcal{L}_{M,C,I,A}$ is very similar to pattern-matching Spi calculus [23] and Psi calculi [1], albeit with structural channel terms, and without the assertions or the possibility of repeated binding names in patterns. There are also similarities between $\mathcal{L}_{M,C,I,A}$ and the polyadic synchronous π-calculus of [6], although the intensionality is limited to the channel, i.e. inputs and outputs of the form $s(\lambda x).P$ and $\overline{s}\langle a\rangle.P$ respectively. With respect to symmetry: $\mathcal{L}_{P,C,NO,CS}$ is closest to the fusion calculus [35] although the scope of binding in communication is different; $\mathcal{L}_{M,D,I,F}$ corresponds to CPC; and $\mathcal{L}_{M,D,I,CS}$, $\mathcal{L}_{M,C,I,CS}$, and $\mathcal{L}_{M,C,I,F}$ to variants of CPC [13].

There are already existing specific results for some symmetric process calculi that agree with the results here. CPC ($\mathcal{L}_{M,D,I,F}$) can homomorphically encode: π-calculus ($\mathcal{L}_{M,C,NO,A}$), Linda ($\mathcal{L}_{P,D,NM,A}$), and Spi Calculus (perhaps $\mathcal{L}_{M,C,I,A}$) while none of them can encode CPC [17, 18, 15]. Meanwhile fusion calculus ($\mathcal{L}_{P,C,NO,CS}$) and Psi calculi ($\mathcal{L}_{M,C,I,A}$) are unrelated to CPC in that neither can encode CPC, and CPC cannot encode either of them [17, 13, 18]. Similarly fusion calculus can encode π-calculi, although not the other way around [35]. Impossibility of encoding results for CPC and fusion calculus into many calculi can be derived from the results here.

C.3 Future Work

Future formal work in this area could include exploring the rôle of either logics or multi-party synchrony in communication. Logics that play a rôle in communication would allow for capturing the behaviours of languages like Concurrent Constraint Programming [37] or Psi calculi. The rôle of logics has been used to show separation of Psi calculi from CPC [16]. Multi-party synchrony would allow for representing the behaviour of languages such as join calculus [11], general rendezvous calculus [3], or m-calculus [38], where arbitrary numbers of processes must coordinate to yield a reduction, not always two. Some of this has begun in [19] although this only accounts for joining and has not been connected to the results here.

References

1 Jesper Bengtson, Magnus Johansson, Joachim Parrow, and Björn Victor. Psi-calculi: a framework for mobile processes with nominal data and logic. *Logical Methods in Computer Science*, 7(1), 2011.

2 Jesper Bengtson and Joachim Parrow. Formalising the pi-calculus using nominal logic. *Logical Methods in Computer Science*, 5(2), 2009.

3 Laura Bocchi and Lucian Wischik. A process calculus of atomic commit. *Electronic Notes in Theoretical Computer Science*, 105(0):119 – 132, 2004. Proceedings of the First International Workshop on Web Services and Formal Methods (WSFM 2004) M. Bravetti and G. Zavattaro.

4 G. Boudol. Notes on algebraic calculi of processes. In Krzysztof R. Apt, editor, *Logics and Models of Concurrent Systems*, pages 261–303. Springer-Verlag New York, Inc., New York, NY, USA, 1985.

5 Nadia Busi, Roberto Gorrieri, and Gianluigi Zavattaro. On the expressiveness of linda coordination primitives. *Information and Computation*, 156(1-2):90–121, 2000.

6 Marco Carbone and Sergio Maffeis. On the expressive power of polyadic synchronisation in π-calculus. *Nordic Journal of Computing*, 10(2):70–98, May 2003.

7 Luca Cardelli and Andrew D. Gordon. Mobile ambients. In *Foundations of Software Science and Computation Structures: First International Conference, FoSSaCS '98*, pages 140–155, 1998.

8 Giuseppe Castagna, Rocco De Nicola, and Daniele Varacca. Semantic subtyping for the pi-calculus. *Theoretical Computer Science*, 398(1-3):217–242, May 2008.

9 Rocco De Nicola, Daniele Gorla, and Rosario Pugliese. On the expressive power of klaim-based calculi. *Theoretical Computer Science*, 356(3):387–421, May 2006.

10 Robert de Simone. Higher-level synchronising devices in Meije-SCCS. *Theoretical Computer Science*, 37:245–267, 1985.

11 Cedric Fournet and Georges Gonthier. The reflexive cham and the join-calculus. In *IN PROCEEDINGS OF THE 23RD ACM SYMPOSIUM ON PRINCIPLES OF PROGRAMMING LANGUAGES*, pages 372–385. ACM Press.

12 David Gelernter. Generative communication in Linda. *ACM Transactions on Programming Languages and Systems*, 7(1):80–112, 1985.

13 Thomas Given-Wilson. *Concurrent Pattern Unification*. PhD thesis, University of Technology, Sydney, Australia, 2012.

14 Thomas Given-Wilson. An Intensional Concurrent Faithful Encoding of Turing Machines. In *7th Interaction and Concurrency Experience (ICE 2014)*, Berlin, Germany, June 2014.

15 Thomas Given-Wilson. On the Expressiveness of Intensional Communication. In *Combined 21th International Workshop on Expressiveness in Concurrency and 11th Workshop on Structural Operational Semantics*, Rome, Italie, September 2014.

16 Thomas Given-Wilson and Daniele Gorla. Pattern matching and bisimulation. In *Proc. of COORDINATION*, volume 7890 of *LNCS*, pages 60–74. Springer, 2013.

17 Thomas Given-Wilson, Daniele Gorla, and Barry Jay. Concurrent pattern calculus. In *Proc. of IFIP-TCS*, volume 323 of *IFIP Advances in Information and Communication Technology*, pages 244–258. Springer, 2010.

18 Thomas Given-Wilson, Daniele Gorla, and Barry Jay. A Concurrent Pattern Calculus. *To appear in: Logical Methods in Computer Science*, 10(3), 2014.

19 Thomas Given-Wilson and Axel Legay. On the Expressiveness of Joining. In *8th Interaction and Concurrency Experience (ICE 2015)*, Grenoble, France, June 2015.

20 D. Gorla. Comparing communication primitives via their relative expressive power. *Information and Computation*, 206(8):931–952, 2008.

21 D. Gorla. A taxonomy of process calculi for distribution and mobility. *Distributed Computing*, 23(4):273–299, 2010.

22 D. Gorla. Towards a unified approach to encodability and separation results for process calculi. *Information and Computation*, 208(9):1031–1053, 2010.

23 Christian Haack and Alan Jeffrey. Pattern-matching spi-calculus. *Information and Computation*, 204(8):1195–1263, August 2006.

24 Bjørn Haagensen, Sergio Maffeis, and Iain Phillips. Matching systems for concurrent calculi. *Electronic Notes in Theoretical Computer Science*, 194(2):85 – 99, 2008. Proceedings of the 14th International Workshop on Expressiveness in Concurrency (EXPRESS 2007).

25 Barry Jay. *Pattern Calculus: Computing with Functions and Data Structures*. Springer, 2009.

26 Barry Jay and Thomas Given-Wilson. A combinatory account of internal structure. *Journal of Symbolic Logic*, 76(3):807–826, 2011.

27 Ivan Lanese, Jorge A. Pérez, Davide Sangiorgi, and Alan Schmitt. On the expressiveness of polyadic and synchronous communication in higher-order process calculi. In *Proc. of ICALP*, volume 6199 of *LNCS*, pages 442–453. Springer, 2010.

28 Ivan Lanese, Cátia Vaz, and Carla Ferreira. On the expressive power of primitives for compensation handling. In *Proceedings of the 19th European Conference on Programming Languages and Systems*, ESOP'10, pages 366–386, Berlin, Heidelberg, 2010. Springer-Verlag.

29 Robin Milner. The polyadic π-calculus: A tutorial. In *Logic and Algebra of Specification*, volume 94 of *Series F*. NATO ASI, Springer, 1993.

30 Robin Milner, Joachim Parrow, and David Walker. A calculus of mobile processes, I. *Information and Computation*, 100(1):1–40, September 1992.

31 Robin Milner, Joachim Parrow, and David Walker. A calculus of mobile processes, II. *Information and Computation*, 100(1):41–77, September 1992.

32 Rocco De Nicola, Gian Luigi Ferrari, and Rosario Pugliese. KLAIM: A kernel language for agents interaction and mobility. *IEEE Transactions on Software Engineering*, 24(5):315–330, 1998.

33 Lasse Nielsen, Nobuko Yoshida, and Kohei Honda. Multiparty symmetric sum types. In *Proceedings of the 17th International Workshop on Expressiveness in Concurrency (EXPRESS 2010)*, pages 121–135, 2010.

34 Catuscia Palamidessi. Comparing the expressive power of the synchronous and asynchronous pi-calculi. *Mathematical. Structures in Comp. Sci.*, 13(5):685–719, October 2003.

35 J. Parrow and B. Victor. The fusion calculus: expressiveness and symmetry in mobile processes. In *Proceedings of Thirteenth Annual IEEE Symposium on Logic in Computer Science*, pages 176–185, Jun 1998.

36 Joachim Parrow. Expressiveness of process algebras. *Electronic Notes in Theoretical Computer Science*, 209:173–186, April 2008.

37 Vijay A. Saraswat, Martin Rinard, and Prakash Panangaden. The semantic foundations of concurrent constraint programming. In *Proceedings of the 18th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, POPL '91, pages 333–352, New York, NY, USA, 1991. ACM.

38 Alan Schmitt and Jean-Bernard Stefani. The m-calculus: a higher-order distributed process calculus. In Alex Aiken and Greg Morrisett, editors, *Conference Record of POPL 2003: The 30th SIGPLAN-SIGACT Symposium on Principles of Programming Languages, New Orleans, Louisisana, USA, January 15-17, 2003*, pages 50–61. ACM, 2003.

39 Christian Urban, Stefan Berghofer, and Michael Norrish. Barendregt's variable convention in rule inductions. In Frank Pfenning, editor, *Automated Deduction – CADE-21*, volume 4603 of *Lecture Notes in Computer Science*, pages 35–50. Springer Berlin Heidelberg, 2007.

40 Rob J. van Glabbeek. Musings on encodings and expressiveness. In *Proceedings of EXPRESS/SOS*, volume 89 of *EPTCS*, pages 81–98, 2012.