On the Expressiveness of Symmetric Communication

Thomas Given-Wilson, Axel Legay

On the Expressiveness of Symmetric Communication

Thomas Given-Wilson and Axel Legay

Inria

Abstract. The expressiveness of communication primitives has been explored in a common framework based on the π-calculus 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 supporting exchange of information between an action and co-action, or as unification of actions. By means of possibility/impossibility of encodings, this paper shows that the exchange approach is related to, or more expressive than, many previously considered languages. Meanwhile, the unification approach is more expressive than some, but mostly unrelated to, other languages.

1 Introduction

The expressiveness of process calculi based upon their communication primitives has been widely explored before [19, 3, 4, 12, 6, 8]. In [12, 8] this is detailed by examining combinations of four features, namely: *synchronism*, *arity*, *communication medium*, and *pattern-matching*. These features are able to represent many popular calculi including: monadic or polyadic π-calculus [17, 18, 16]; Linda [5]; asymmetric variations of Concurrent Pattern Calculus (CPC) [10, 11, 6, 7]; and Psi calculi [1]. However, all these calculi exploit upon asymmetric input and output behaviour.

Symmetric behaviour has been considered before in process calculi. One example, fusion calculus [20] 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[10, 11].

This paper abstracts away from specific calculi in the style of [12, 8] 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 paradigms, while also generalising to something that can be applied to any calculus. However, there is some complexity when deciding how this should be represented in an abstract calculus since there are two reasonable choices.

The first choice is to consider symmetry to support *exchange*, where an action and co-action interact and allow both input and output from either side. For example, buying a paper online could be represented by two processes, a buyer $B = n(\lambda p, \texttt{Credit}).B'$ that desires a paper (to be bound to) p and has credit card information `Credit`; and a journal $J = \overline{n}\langle \texttt{Paper}, \lambda c\rangle.J'$ that will sell the paper `Paper` in exchange for credit card

information (bound to) c. These processes can then communicate along the channel n to perform an exchange in a single interaction by:

$$B \mid J \;=\; n(\lambda p, \texttt{Credit}).B' \mid \overline{n}\langle \texttt{Paper}, \lambda c\rangle.J' \;\longmapsto\; \{\texttt{Paper}, p\}B' \mid \{\texttt{Credit}, c\}J'$$

where the buyer now has `Paper` bound to p in their continuation B', and the journal has `Credit` bound to c in S'. This exchange approach of action and co-action with both input and output on both sides aligns with the fusion calculus style of interaction.

The second choice is to consider symmetry to support symmetric *unification*, where a single communication primitive is used for interaction. This style supports self recognition, e.g. the self recognising process $S = a(a).S'$ that does not reduce, but when in parallel composition with itself it can interact as below:

$$S \mid S \;=\; a(a).S' \mid a(a).S' \;\longmapsto\; S' \mid S'$$

since the channel name a is the same and the structure a can by unified with a. Observe that exchange style (above) can also be represented with this style by changing the co-action (or journal) to an action $J = n(\texttt{Paper}, \lambda c).J'$ with interaction occurring as before. This approach of a symmetric unification via a single interaction primitive and allowing self recognition (as well as exchange) aligns with CPC style interaction.

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.

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

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

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

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

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

By extending prior work with the new dimension of symmetry (and replacing synchronism since all exchange and unification languages must be synchronous), the original twelve calculi of [8] are here expanded to thirty-six. 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. In general exchange languages are more expressive than their asymmetric counterparts. There are a couple of exceptions that can exploit a lack of pattern-matching to encode sufficient information to control interaction, but these are the exceptions. Most exchange languages can be encoded into asymmetric languages that support more matching of names, the exceptions are when the exchange languages is polyadic and supports some kind of name-matching. Within the exchange languages expressiveness increases in the same manner as asymmetric

languages, although the interplay of polyadicity and intensionality is slightly more subtle and does not support the same equivalences. No unification language can be encoded into an exchange or asymmetric language, this is due to the self recognition process S. Within unification languages expressiveness increases in exactly the same manner as asymmetric languages. Unification languages do not require name-matching to be able to determine equivalence of names. This allows unification languages without name-matching to encode all asymmetric and exchange languages with name matching (but not intensionality). Finally, all the intensional unification languages are equally expressive, and are the most expressive languages considered here.

These results explore two approaches to symmetric communication and expressiveness when combined with other features. This captures many published calculi, as well as providing characterisations of others. 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 considered calculi. Section 3 revises the encoding criteria used for comparing calculi. Section 4 provides a diagrammatic overview of the results. Section 5 explores new relations between and within asymmetric and exchange languages. Section 6 considers unification languages and their relations. Section 7 concludes.

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. With the exception of the symmetric constructs this is similar to prior definitions from [8].

Assume a countable set of names $\mathcal{N}$ (denoted a, b, c). *Name-matching patterns* (denoted m, n, o), and *symmetric patterns* (denoted p, q) are defined by:

$$\begin{array}{rcll} m, n & ::= & \lambda x & \textit{binding name} \\ & | & \ulcorner a \urcorner & \textit{name-match} \end{array} \qquad \begin{array}{rcll} p, q & ::= & a & \textit{name} \\ & | & m & \textit{name-match patterns} \\ & | & p \bullet q & \textit{compound}. \end{array}$$

Binding names (denoted $\lambda x, \lambda y, \lambda z$) are used to indicate input behaviour, name-matches $\ulcorner a \urcorner$ test for equality, and compounds combine to symmetric patterns into one (all as in [11, 8]). 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 linear iff all binding names within the pattern are pairwise distinct. The rest of this paper will only consider linear input patterns.

The symmetric patterns here both; generalise concepts in similar work, and are too general for some of the languages here. Thus define two subsets of the symmetric patterns. The *terms* (denoted s, t) are the symmetric patterns that contain no binding names or matched names. (These correspond to the terms of [8], the communicable patterns of CPC, and the output structures of Psi calculi.) The *intensional patterns* (denoted f, g) are the symmetric patterns that contain no names, i.e. they entirely of name-matching patterns and compounds. (These correspond to the intensional patterns of [8].)

The (parametric) syntax for the languages is:

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

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: $\mathbf{0}$ denotes the null process; restriction $(\nu n)P$ restricts the visibility of n to P; and parallel composition $P|Q$ allows independent evolution of P and Q. The **if** $s = t$ **then** P **else** Q represents conditional equivalence with **if** $s = t$ **then** P used when Q is $\mathbf{0}$ (like the name match of π-calculus, **if** $s = t$ **then** P **else** Q blocks either P when $s \neq t$ or Q when $s = t$). The $*P$ represents replication of the process P. Finally, the $\surd$ is used to represent a success process or state, exploited for reasoning about encodings as in [14, 6].

This paper considers the possible combinations of four features for communication: *arity* (monadic vs polyadic data), *communication medium* (message passing vs shared dataspaces), *pattern-matching* (simple binding vs name equality vs intensionality), and *symmetry* (asymmetric vs exchange vs unification). 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, E for exchange communication, and U for unification communication.

For simplicity a dash $-$ is 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 three lines define the components of communication primitives based upon the pattern-matching of the language; with *input patterns* IN, *output patterns* OUT, combined patterns ALL, and channel structures CH. The rest defines the languages by their action ACT and co-action $COACT$ using the communication primitives. Observe that for: asymmetric languages actions only contain input patterns, and co-actions contain output patterns; exchange languages action and co-action both contain combined patterns; and the unification languages have a single action (by defining the co-action to be the same) containing combined patterns. Here the denotation $\widetilde{\cdot}$ represents a sequence of the form $\cdot_1, \cdot_2, \ldots, \cdot_n$ and can be used for names, binding names, terms, and both kinds of patterns. As usual $(\nu x)P$ and binding names λx in any form (including IN and ALL) bind x in P. The corresponding notions of free and bound names of a process, denoted $\mathsf{fn}(P)$ and $\mathsf{bn}(P)$, are as usual. Also note that α-conversion (denoted $=_\alpha$) is assumed in the usual manner. An action or co-action is linear if all binding names occur exactly once; this paper shall only consider linear actions and co-actions. Finally, the structural equivalence relation $\equiv$ is defined as follows:

$$P \mid Q \equiv Q \mid P \qquad P \mid (Q \mid R) \equiv (P \mid Q) \mid R \qquad P \equiv P' \quad \text{if } P =_\alpha P'$$
$$\textbf{if } s = t \textbf{ then } P \textbf{ else } Q \equiv P \quad s = t \qquad P \mid \mathbf{0} \equiv P \qquad (\nu a)\mathbf{0} \equiv \mathbf{0}$$
$$(\nu a)(\nu b)P \equiv (\nu b)(\nu a)P \qquad P \mid (\nu a)Q \equiv (\nu a)(P \mid Q) \quad \text{if } a \notin \mathsf{fn}(P) .$$

Most of the asymmetric languages correspond to the communication paradigm of popular process calculi, including (but not limited to): monadic or polyadic π-calculus;

$$
\begin{array}{rlll}
\mathcal{L}_{-,-,NO,-} : & IN ::= \lambda x \quad OUT ::= a & ALL ::= \lambda x \mid a & CH ::= a \\
\mathcal{L}_{-,-,NM,-} : & IN ::= m \quad OUT ::= a & ALL ::= m \mid a & CH ::= a \\
\mathcal{L}_{-,-,I,-} : & IN ::= f \quad OUT ::= t & ALL ::= p & CH ::= t \\
\mathcal{L}_{M,D,-,A} : & ACT ::= (IN) & & COACT ::= \langle OUT \rangle \\
\mathcal{L}_{M,C,-,A} : & ACT ::= CH(IN) & & COACT ::= \overline{CH}\langle OUT \rangle \\
\mathcal{L}_{P,D,-,A} : & ACT ::= (\widetilde{IN}) & & COACT ::= \langle \widetilde{OUT} \rangle \\
\mathcal{L}_{P,C,-,A} : & ACT ::= CH(\widetilde{IN}) & & COACT ::= \overline{CH}\langle \widetilde{OUT} \rangle \\
\mathcal{L}_{M,D,-,E} : & ACT ::= (ALL) & & COACT ::= \langle ALL \rangle \\
\mathcal{L}_{M,C,-,E} : & ACT ::= CH(ALL) & & COACT ::= \overline{CH}\langle ALL \rangle \\
\mathcal{L}_{P,D,-,E} : & ACT ::= (\widetilde{ALL}) & & COACT ::= \langle \widetilde{ALL} \rangle \\
\mathcal{L}_{P,C,-,E} : & ACT ::= CH(\widetilde{ALL}) & & COACT ::= \overline{CH}\langle \widetilde{ALL} \rangle \\
\mathcal{L}_{M,D,-,U} : & ACT, COACT ::= (ALL) & & \\
\mathcal{L}_{M,C,-,U} : & ACT, COACT ::= CH(ALL) & & \\
\mathcal{L}_{P,D,-,U} : & ACT, COACT ::= (\widetilde{ALL}) & & \\
\mathcal{L}_{P,C,-,U} : & ACT, COACT ::= CH(\widetilde{ALL}) \ . & &
\end{array}
$$

Fig. 1. Languages in this paper.

Linda; asymmetric variations of CPC; and Psi calculi. For details on these and other languages see [12, 8]. With respect to symmetry: $\mathcal{L}_{P,C,NO,E}$ is closest in communication paradigm to the fusion calculus [20] although the scope of binding in communication is different. $\mathcal{L}_{M,D,I,U}$ corresponds to the communication paradigm of CPC; and $\mathcal{L}_{M,D,I,E}$, $\mathcal{L}_{M,C,I,E}$, and $\mathcal{L}_{M,C,I,U}$ to the communication paradigms of variants of CPC [6].

Remark 1. Most of the languages can be ordered; in particular $\mathcal{L}_{\alpha_1,\beta_1,\gamma_1,\delta_1}$ is a sub-language of $\mathcal{L}_{\alpha_2,\beta_2,\gamma_2,\delta_2}$ 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 the following axioms:

$$M \leq P \qquad D \leq C \qquad NO \leq NM \leq I \qquad A \leq E \ .$$

This can be understood as a limited language variation being a special case of a more general 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. Asymmetric communication is exchange with only input patterns in actions, and only output patterns in co-actions; and exchange languages are unification 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 [16, 15, 8]. An alternative style is via a *labelled transition system* (LTS) such as [12]. Here the reduction based style is chosen for simplicity. The LTS style can be used for intensional and symmetric languages [1, 6], and indeed captures many of the languages here [9].

Substitutions, denoted σ, ρ in non-pattern-matching and name-matching languages are mappings (with finite domain) from names to names. For intensional languages substitutions are mappings from names to terms. The application of a substitution σ to

a pattern p is defined as follows:

$$\sigma x = \begin{cases} \sigma(x) & x \in \text{domain}(\sigma) \\ x & x \notin \text{domain}(\sigma) \end{cases} \qquad \sigma\ulcorner x\urcorner = \ulcorner(\sigma x)\urcorner \qquad \sigma(p \bullet q) = (\sigma p) \bullet (\sigma q)\ .$$

Where substitution is as usual on names, and on the understanding that the name-match syntax can be applied to any term as follows: $\ulcorner x\urcorner \stackrel{\text{def}}{=} \ulcorner x\urcorner$ and $\ulcorner(s \bullet t)\urcorner \stackrel{\text{def}}{=} \ulcorner s\urcorner \bullet \ulcorner t\urcorner$. Given a substitution σ and a process P, denote with σP the usual capture-avoiding application of σ to P. As usual capture can be avoided by exploiting α-equivalence [21, 2].

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

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

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 linearity of patterns, and this holds for the following two rules also.

The asymmetric and exchange 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 below:

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

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

The empty sequence matches with the empty sequence 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, such as when; when a single unification fails, a term is aligned with a term, an intensional pattern with an intensional pattern, or when the sequences are of unequal arity.

The unification 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:

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

Interaction is now defined by the following two axioms. The first

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

for asymmetric and exchange languages; and the second

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

for the unification languages. In both the s's are omitted for 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 $\cong$ denote a reduction-sensitive reference behavioural equivalence for that language, e.g. a barbed equivalence. That is, a behavioural equivalence $\cong$ such that whenever $P \cong P'$ and $P' \longmapsto$ imply $P \longmapsto$ as in Definition 5.3 of [14] (observe that his rules out weak bisimulations for example). For the asymmetric languages these are already known, either by their equivalent language in the literature or from [12, 9, 8]. For the non-asymmetric languages the results in [9] can be applied.

3 Encodings

This section recalls the definition of valid encodings as well as some useful theorems (details in [14]) for formally relating process calculi. The choice of valid encodings here is to align with prior works [12, 14, 8] and where possible reuse prior results. Further discussion of this choice is provided in Appendix B.2.

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 (of the source language) into a tuple of k names (of the target language), for $k > 0$. In doing this, the translation may fix some names to play a precise rôle or may translate a single name into a tuple of names, this can be obtained by exploiting $\varphi_{\llbracket\,\rrbracket}$.

Now consider only encodings that satisfy the following properties. Let a *k-ary context* $(\cdot_1;\ldots;\cdot_k)$ be a process with k holes. Denote with $\longmapsto^\omega$ an infinite sequence of reductions and let $P \Downarrow$ mean there exists P' such that $P \Longmapsto P'$ and $P' \equiv P'' \mid \surd$ for some P''. Moreover, let $\cong$ denote the reference behavioural equivalence. Finally, to simplify reading, let S range over processes of the source language (viz., $\mathcal{L}_1$) and T range over processes of the target language (viz., $\mathcal{L}_2$).

Definition 1 (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* $\mathcal{C}^N_{\mathsf{op}}(\cdot_1;\ldots;\cdot_k)$ *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* $\llbracket\, \mathsf{op}(S_1,\ldots,S_k)\,\rrbracket = \mathcal{C}^N_{\mathsf{op}}(\llbracket\, S_1\,\rrbracket;\ldots;\llbracket\, S_k\,\rrbracket)$.
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* $\llbracket\,\sigma S\,\rrbracket \cong_2 \sigma'\llbracket\, S\,\rrbracket$ *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* $\llbracket\, S\,\rrbracket \Longmapsto_2 \cong_2 \llbracket\, S'\,\rrbracket$;
   - *for all* $\llbracket\, S\,\rrbracket \Longmapsto_2 T$, *there exists* S' *such that* $S \Longmapsto_1 S'$ *and* $T \Longmapsto_2 \cong_2 \llbracket\, S'\,\rrbracket$.
4. Divergence reflection: *for every* S *such that* $\llbracket\, S\,\rrbracket \longmapsto_2^\omega$, *it holds that* $S \longmapsto_1^\omega$.
5. Success sensitiveness: *for every* S, *it holds that* $S \Downarrow_1$ *if and only if* $\llbracket\, S\,\rrbracket \Downarrow_2$.

The following are here recalled from prior works as they are useful for later proofs.

Proposition 1 (Proposition 5.5 from [14]). *Let* $\llbracket\cdot\rrbracket$ *be a valid encoding; then,* $S \mapsto\!\!\!\!\!\!\rightarrow$ *implies that* $\llbracket\, S\,\rrbracket \mapsto\!\!\!\!\!\!\rightarrow$.

Proposition 2 (Proposition 5.6 from [14]). *Let* $\llbracket\cdot\rrbracket$ *be a valid encoding; then for every set of names N, it holds that* $\mathcal{C}^N_{|}(\cdot_1,\cdot_2)$ *has both its holes at top-level.*

Proposition 3 (Proposition 5.7 from [14]). *Let* $\llbracket\cdot\rrbracket$ *be a valid encoding; if there exist two processes* S_1 *and* S_2 *such that* $S_1 \mid S_2 \Downarrow$, *with* $S_i \not\Downarrow$ *and* $S_i \mapsto\!\!\!\!\!\!\rightarrow$ *for* $i = 1, 2$, *then* $\llbracket\, S_1\,\rrbracket \mid \llbracket\, S_2\,\rrbracket \longmapsto$.

The following proof exploits the *matching degree* of a language $\textsc{Md}(\cdot)$, defined as the least upper bound on the number of names that can be matched to yield reduction.

Theorem 1 (5.9 from [14]). *If* $\textsc{Md}(\mathcal{L}_1) > \textsc{Md}(\mathcal{L}_2)$ *then there is no valid encoding of* $\mathcal{L}_1$ *into* $\mathcal{L}_2$.

Theorem 2 (5.8 from [14]). *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* $\llbracket\cdot\rrbracket$ *from* $\mathcal{L}_1$ *to* $\mathcal{L}_2$.

The general way to prove the lack of a valid encoding is done as follows. By contradiction assuming there is a valid encoding $\llbracket\cdot\rrbracket$. Find a pair of processes P and Q the satisfy Proposition 3 such that $P \mid Q \longmapsto$ and $\llbracket\, P \mid Q\,\rrbracket \longmapsto$. From Q obtain some Q' such that $P \mid Q' \mapsto\!\!\!\!\!\!\rightarrow$ and $\llbracket\, P \mid Q'\,\rrbracket \longmapsto$. Conclude by showing this in contradiction with some properties of the encoding or one of the propositions above.

The following result is a consequence of the choices of languages and encoding criteria, which corresponds to formalising Remark 1.

Theorem 3. *If a language* $\mathcal{L}_1$ *is a sub-language of* $\mathcal{L}_2$ *then there exists a valid encoding* $\llbracket\cdot\rrbracket$ *from* $\mathcal{L}_1$ *into* $\mathcal{L}_2$.

Finally, the existence of encodings $\llbracket\cdot\rrbracket_1$ from $\mathcal{L}_1$ into $\mathcal{L}_2$ and $\llbracket\cdot\rrbracket_2$ from $\mathcal{L}_2$ into $\mathcal{L}_3$ does not ensure that $\llbracket\,\llbracket\cdot\rrbracket_1\,\rrbracket_2$ is a valid encoding from $\mathcal{L}_1$ into $\mathcal{L}_3$ [13]. However, this does hold when the encodings use $\equiv$ rather than $\cong$ in the target language, as is the case for all encodings presented in this work. This allows later assumption of composition of encodings here, although this is not true for all valid encodings in general.

4 Overview of Results

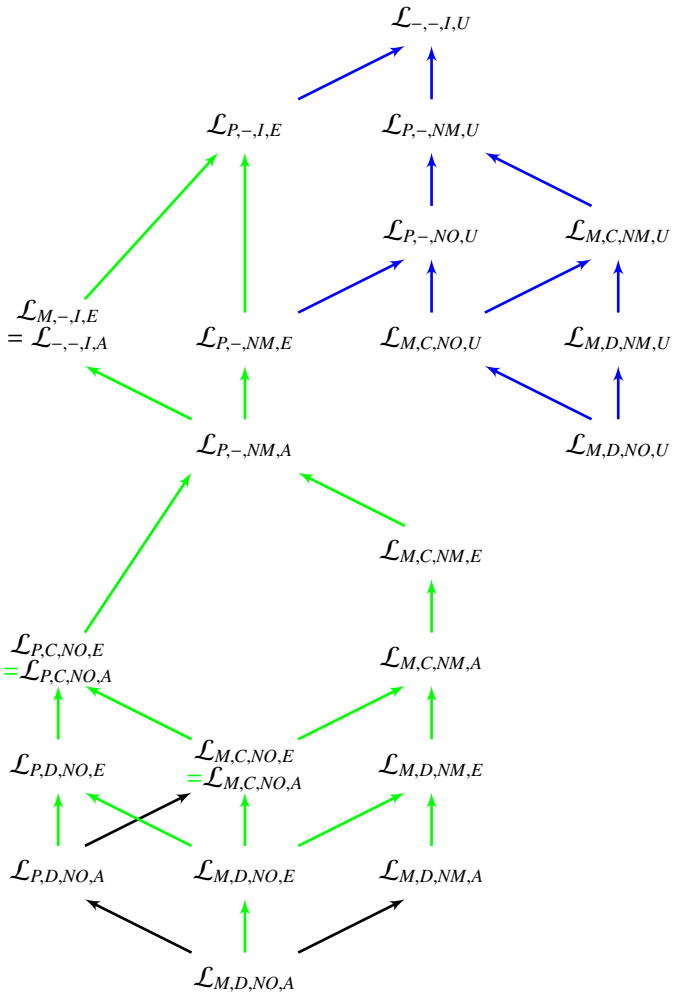

Fig. 2. Relations between all languages

A diagram illustrating the results can be seen in Figure 2. Arrows show increased expressive power and ='s show equivalence; black are from prior work, green from Section 5, and blue from Section 6. The lack of an arrow indicates no possible encoding in either direction (e.g. between $\mathcal{L}_{P,-,I,E}$ and $\mathcal{L}_{P,-,NM,U}$). Transitive relations are omitted (e.g. $\mathcal{L}_{P,-,NM,A}$ to $\mathcal{L}_{P,-,NO,U}$).

5 Asymmetry and Exchange

Exchange is a generalisation of asymmetric communication, i.e. $\mathcal{L}_{\alpha,\beta,\gamma,A}$ is trivially encoded by, $\mathcal{L}_{\alpha,\beta,\gamma,E}$ by Theorem 3. The rest of this section details other relations between asymmetric and exchange languages.

5.1 Exchange in Monadic Non-Intensional Languages

This section considers the simpler languages and demonstrates the proof techniques to show that exchange cannot be easily encoded into asymmetry.

For the monadic non-intensional languages changing from asymmetric to exchange communication alone is almost always an increase in expressive power. The following three results appear similar, but rely on increasingly advanced proof techniques to demonstrate and so are presented separately.

Theorem 4. *There exists no valid encoding of* $\mathcal{L}_{M,D,NO,E}$ *into* $\mathcal{L}_{M,D,NO,A}$.

Theorem 5. *There exists no valid encoding of* $\mathcal{L}_{M,D,NM,E}$ *into* $\mathcal{L}_{M,D,NM,A}$.

Theorem 6. *There exists no valid encoding of* $\mathcal{L}_{M,C,NM,E}$ *into* $\mathcal{L}_{M,C,NM,A}$.

The exception to the general $\mathcal{L}_{M,\beta,\gamma,E}$ is more expressive than $\mathcal{L}_{M,\beta,\gamma,A}$ when $\gamma \neq I$ is $\mathcal{L}_{M,C,NO,E}$ into $\mathcal{L}_{M,C,NO,A}$. This is detailed in Section 5.2.

Within the monadic non-intensional exchange languages the usual diamond of relations exists where adding channel-based communication or name-matching are both increases in expressive power. The separation results between $\mathcal{L}_{M,D,NM,E}$ and $\mathcal{L}_{M,C,NO,E}$ is the most interesting result, as the rest can be proved via matching degree.

Theorem 7. *The languages* $\mathcal{L}_{M,D,NM,E}$ *and* $\mathcal{L}_{M,C,NO,E}$ *are unrelated.*

5.2 Encoding Exchange into Asymmetry

This section considers where exchange languages can be encoded by asymmetric languages. Note that this does not ensure atomicity that motivates some languages [6].

An exchange 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: $\mathcal{L}_1$ and $\mathcal{L}_2$ are both channel-based no-matching languages; or $\mathcal{L}_2$ has a greater matching degree and is polyadic or channel-based.

In the first case, the key idea is to represent the channel name by a pair of names to indicate whether the input is on the action or co-action. Consider the following translation from $\mathcal{L}_{M,C,NO,E}$ into $\mathcal{L}_{M,C,NO,A}$:

$$
\begin{aligned}
[\![(\nu n)P]\!] &\stackrel{\text{def}}{=} (\nu n_1)(\nu n_2)[\![P]\!] \\
[\![n(\lambda x).P]\!] &\stackrel{\text{def}}{=} n_1(\lambda \mathsf{rn}).\mathsf{rn}(\lambda x_1).x(\lambda x_2).[\![P]\!] \\
[\![\overline{n}\langle \lambda x\rangle.P]\!] &\stackrel{\text{def}}{=} n_2(\lambda \mathsf{rn}).\mathsf{rn}(\lambda x_1).x(\lambda x_2).[\![P]\!] \\
[\![n(a).P]\!] &\stackrel{\text{def}}{=} (\nu \mathsf{rn})\overline{n_2}\langle \mathsf{rn}\rangle.\overline{\mathsf{rn}}\langle a_1\rangle.\overline{\mathsf{rn}}\langle a_2\rangle.[\![P]\!] \\
[\![\overline{n}\langle a\rangle.P]\!] &\stackrel{\text{def}}{=} (\nu \mathsf{rn})\overline{n_1}\langle \mathsf{rn}\rangle.\overline{\mathsf{rn}}\langle a_1\rangle.\overline{\mathsf{rn}}\langle a_2\rangle.[\![P]\!] \\
[\![\textbf{if } s = t \textbf{ then } P \textbf{ else } Q]\!] &\stackrel{\text{def}}{=} \textbf{if } s_1 = t_1 \textbf{ then } [\![P]\!] \textbf{ else } [\![Q]\!] \ .
\end{aligned}
$$

Here the names n_1 and n_2 represent two parts of the name n, and rn is a reserved name, these are all introduced by the renaming policy $\varphi_{[\![\,]\!]}$ [14, 10].

Theorem 8. *The encoding from $\mathcal{L}_{M,D,NO,E}$ into $\mathcal{L}_{M,C,NO,A}$ is valid.*

The above encoding illustrates how channel-based communication is sufficient when no name-matching or intensionality is included in the source language.

In the second case, 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 1. The clearest illustration of this when the source language is monadic is the following encoding from $\mathcal{L}_{M,D,NO,E}$ 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 that are translated as follows:

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

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

Theorem 9. *The encoding from $\mathcal{L}_{M,D,NO,E}$ into $\mathcal{L}_{M,C,NO,A}$ is valid.*

Theorem 10. *There exists no valid encoding from $\mathcal{L}_{M,C,NO,A}$ into $\mathcal{L}_{M,D,NO,E}$.*

The following results are achieved in the same manner by extending the initial translation to consider name matches $\ulcorner a \urcorner$ to also be inputs, e.g. i.e. $[\![(\ulcorner a \urcorner).P]\!] \stackrel{\text{def}}{=} \mathsf{ia}(\ulcorner a \urcorner).[\![P]\!]$.

Theorem 11. *The encoding from $\mathcal{L}_{M,D,NM,E}$ into $\mathcal{L}_{M,C,NM,A}$ is valid.*

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

The below result is similar in technique to Theorem 11, instead of the reserved names being used as a channel they are simply added as another part of the polyadic input or output (with name-matching on the input). The following separation result is achieved via matching degree.

Theorem 13. *There exist valid encodings from $\mathcal{L}_{M,C,NM,E}$ into $\mathcal{L}_{P,-,NM,A}$.*

Theorem 14. *There exists no valid encoding from $\mathcal{L}_{P,-,NM,A}$ into $\mathcal{L}_{M,C,NM,E}$.*

A similar but more complex technique can be used to encode polyadic no-matching exchange languages into asymmetric languages. This is illustrated by the following encoding from $\mathcal{L}_{P,D,NO,E}$ 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 names and binding names into a bit-string and also the *complement* (or bitwise not) $\textsc{Not}(\cdot)$ of bit-strings (where ';' is concatenation):

$$\begin{array}{lll} \textsc{Bin}(a) = \mathtt{0} & \textsc{Bin}(\lambda x) = \mathtt{1} & \textsc{Bin}(n, \widetilde{n}) = \textsc{Bin}(n); \textsc{Bin}(\widetilde{n}) \\ \textsc{Not}(\mathtt{0}) = \mathtt{1} & \textsc{Not}(\mathtt{1}) = \mathtt{0} & \textsc{Not}(\mathtt{X}, \widetilde{\mathtt{X}}) = \textsc{Not}(\mathtt{X}); \textsc{Not}(\widetilde{\mathtt{X}}) \ . \end{array}$$

Given a sequence of binding names and names $\widetilde{p}$, the sequences of the binding names $\textsc{Bn}(\widetilde{p})$, and names $\textsc{Nm}(\widetilde{p})$ are defined by:

$$\begin{array}{ll} \textsc{Bn}(\lambda x, \widetilde{p}) = \lambda x, \textsc{Bn}(\widetilde{p}) & \textsc{Nm}(\lambda x, \widetilde{p}) = \textsc{Nm}(\widetilde{p}) \\ \textsc{Bn}(a, \widetilde{p}) \ = \textsc{Bn}(\widetilde{p}) & \textsc{Nm}(a, \widetilde{p}) \ = a, \textsc{Nm}(\widetilde{p}) \ . \end{array}$$

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:

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

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 to complete the interaction.

Theorem 15. *The encoding from $\mathcal{L}_{P,D,NO,E}$ into $\mathcal{L}_{P,C,NO,A}$ is valid.*

Theorem 16. *There exists no valid encoding from $\mathcal{L}_{P,C,NO,A}$ into $\mathcal{L}_{P,D,NO,E}$.*

The encoding from $\mathcal{L}_{P,C,NO,E}$ into $\mathcal{L}_{P,C,NO,A}$ exploits elements of the technique above. Define the function $\textsc{Val}(\cdot)$ that gives the numeric value of a binary string, e.g. $\textsc{Val}(\mathtt{101}) = 5$ and $\textsc{Val}(\mathtt{1010}) = 10$. Now the encoding from $\mathcal{L}_{P,C,NO,E}$ into $\mathcal{L}_{P,C,NO,A}$ can be constructed as follows exploiting a reserved name rn as usual:

$$\begin{array}{l} [\![\, n(\widetilde{p}).P \,]\!] \stackrel{\text{def}}{=} n(\lambda \mathsf{rn}, \textsc{Bn}(\widetilde{p}), \lambda z, \ldots, \lambda z_i).\overline{\mathsf{rn}}\langle \textsc{Nm}(\widetilde{p}) \rangle.[\![\, P \,]\!] \\ \qquad\qquad \text{where } i = \textsc{Val}(1; \textsc{Bin}(\widetilde{p})) - |\textsc{Bn}(\widetilde{p})| \\ [\![\, \overline{n}\langle \widetilde{p} \rangle.P \,]\!] \stackrel{\text{def}}{=} (\nu \mathsf{rn})(\nu z_1) \ldots (\nu z_i) \overline{n}\langle \mathsf{rn}, \textsc{Nm}(\widetilde{p}), z_1, \ldots, z_i \rangle.\mathsf{rn}(\textsc{Bn}(\widetilde{p})).[\![\, P \,]\!] \\ \qquad\qquad \text{where } i = \textsc{Val}(1; \textsc{Not}(\textsc{Bin}(\widetilde{p}))) - |\textsc{Nm}(\widetilde{p})| \end{array}$$

and translating all other processes homomorphically. Also $\widetilde{z}$ do not intersect one another, or any of the names in n and $\widetilde{p}$ and $fn(P)$.

The key idea is to map the binary representation of the structure of the action or co-action to the arity of the encoded action or co-action. To prevent conflicts between encodings, for example $n(a, \lambda x)$ and $n(\lambda x)$, the binary representation is pre-pended with 1. Thus, the arity of the action or co-action ensures correct interaction if the structure is correct, and the channel name is matched as usual.

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

Building on Theorem 15 and the equivalence between the languages $\mathcal{L}_{P,-,NM,A}$ [12] conclude that $\mathcal{L}_{P,-,NM,A}$ are able to encode all the: monadic non-intensional exchange languages; and the polyadic no-matching exchange languages.

5.3 Other Relations With Bounded Matching Degree

This section considers other relations between languages equally or less expressive than $\mathcal{L}_{P,-,NM,A}$, i.e. all the languages that can be encoded in $\mathcal{L}_{P,-,NM,A}$.

Within exchange languages, clearly $\mathcal{L}_{M,\beta,NO,E}$ is a sub-language of $\mathcal{L}_{P,\beta,NO,E}$ for any β and so can be validly encoded by Theorem 3. The following proves the separation results required to indicate an increase in expressiveness.

Theorem 18. *There exists no valid encoding of $\mathcal{L}_{P,D,NO,E}$ into $\mathcal{L}_{M,D,NO,E}$.*

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

Relating to asymmetric languages, $\mathcal{L}_{P,D,NO,A}$ is a sub-language of $\mathcal{L}_{P,D,NO,E}$ and can be validly encoded by Theorem 3. The following proves an increase in expressiveness.

Theorem 20. *There exists no valid encoding of $\mathcal{L}_{P,D,NO,E}$ 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 these results.

The intensional asymmetric languages all have equivalent expressiveness (by Theorem 6.5 of [8]) and to the monadic exchange languages. Consider the languages $\mathcal{L}_{M,D,I,E}$ and $\mathcal{L}_{M,C,I,E}$, there is a trivial valid encoding of $\mathcal{L}_{M,D,I,E}$ into $\mathcal{L}_{M,C,I,E}$ by Theorem 3. The following shows equivalence via the reverse encoding from $\mathcal{L}_{M,C,I,E}$ into $\mathcal{L}_{M,D,I,E}$. 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):

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

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

The translation compounds the channel pattern p with the term or intensional pattern q, converting to maintain being either a term or intensional pattern.

Theorem 21. *The encoding from $\mathcal{L}_{M,C,I,E}$ into $\mathcal{L}_{M,D,I,E}$ is valid.*

The following completes the equivalences. Since all the languages $\mathcal{L}_{-,-,I,A}$ are equally expressive and since the languages $\mathcal{L}_{M,-,I,E}$ are equally expressive by Theorem 21 it suffices to consider examples from either group. The encodings from $\mathcal{L}_{-,-,I,A}$ into $\mathcal{L}_{M,-,I,E}$ follow by $\mathcal{L}_{M,D,I,A}$ being a sub-language of $\mathcal{L}_{M,D,I,E}$. The other direction follows from the below result via a straightforward adaption of Theorem 9.

Theorem 22. *There is a valid encoding from $\mathcal{L}_{M,D,I,E}$ into $\mathcal{L}_{M,C,I,A}$.*

Considering polyadic non-intensional languages, $\mathcal{L}_{P,D,NM,E}$ can be encoded into $\mathcal{L}_{P,C,NM,E}$ by Theorem 3. For the converse, consider the standard approach [12, 8] that is the homeomorphic on all forms except the action and co-action which are as follows:

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

Theorem 23. *The encoding from $\mathcal{L}_{P,C,NM,E}$ into $\mathcal{L}_{P,D,NM,E}$ is valid.*

For the polyadic intensional languages, there is a trivial valid encoding of $\mathcal{L}_{P,D,I,E}$ into $\mathcal{L}_{P,C,I,E}$ by Theorem 3. The following is in the same manner as Theorem 23.

Theorem 24. *The exists a valid encoding from $\mathcal{L}_{P,C,I,E}$ into $\mathcal{L}_{P,D,I,E}$.*

5.5 Concluding Relations

This section concludes the relations between asymmetric and exchange languages by formalising those between languages with unbounded matching degree. In general this is showing separation results between different language groups.

Polyadic intensional exchange languages are more expressive than any other exchange or asymmetric languages. By Theorems 23 & 24 in all languages considered here being dataspace-based or channel-based is immaterial to expressive power. The languages $\mathcal{L}_{P,-,NM,E}$ are sub-languages of $\mathcal{L}_{P,-,I,E}$ and so their expressiveness is included naturally, the reverse is from the following result.

Theorem 25. *There exists no valid encoding of $\mathcal{L}_{P,-,I,E}$ into $\mathcal{L}_{P,-,NM,E}$.*

Comparing with other intensional exchange languages, $\mathcal{L}_{M,-,I,E}$ can be encoded by $\mathcal{L}_{P,-,I,E}$ via Theorem 3. Separation is proved as in Theorem 29.

Theorem 26. *There exists no valid encoding from $\mathcal{L}_{P,-,I,E}$ into $\mathcal{L}_{M,-,I,E}$.*

That the languages $\mathcal{L}_{M,-,I,E}$ are unrelated to $\mathcal{L}_{P,-,NM,E}$ follows from the below two results. Note the groups of languages can be treated equivalently due to Theorems 22 & 23. The proof techniques below are the same as Theorems 25 & 18, respectively.

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

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

Lastly, the languages $\mathcal{L}_{P,-,NM,A}$ can be encoded $\mathcal{L}_{P,-,NM,E}$ via Theorem 3. The reverse is prevented by the following result.

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

6 Unification

This section considers the expressiveness of unification languages, and their relations to asymmetric and exchange languages.

6.1 Unification Cannot be Simulated

The following results show that no unification language can be encoded into an asymmetric or exchange language. Key is a *self recognising* process, defined to be $S = (a).\surd$ for the dataspace-based languages and $S = a(a).\surd$ for the channel-based languages, that has the behaviour $S \mid S \longmapsto\Downarrow$ but $S \mapsto\not\rightarrow$ and $S \not\Downarrow$. This can be exploited since no non-unification process can reduce in parallel with itself unless it reduces alone.

Theorem 30. *There exists no valid encoding of a unification language $\mathcal{L}_{-,-,-,U}$ into any non-unification language $\mathcal{L}_{-,-,-,\delta}$ $\delta \neq U$.*

The above result can be used to prove a separation result from any unification language to a non-unification language, these results are omitted from the rest of the paper.

6.2 On Monadic Non-Intensional Unification Languages

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

The shift to unification leads to $\mathcal{L}_{M,D,NO,U}$ being unrelated to any other monadic, dataspace-based, non-matching language $\mathcal{L}_{M,D,NO,-}$. The proof technique for Theorem 31 can also be used for Theorem 32.

Theorem 31. *There exists no valid encoding from $\mathcal{L}_{M,D,NO,\delta}$ where $\delta \neq U$ into $\mathcal{L}_{M,D,NO,U}$.*

Theorem 32. *There exists no valid encoding from $\mathcal{L}_{M,-,\gamma,\delta}$ into $\mathcal{L}_{M,-,\gamma,U}$ where $\gamma \leq NM$ and $\delta \neq U$.*

The relations between the monadic non-intensional unification languages are as usual, although the usual proof techniques do not always hold. In particular, no-matching unification languages still have non-zero matching degree, so separation results that rely on matching degree alone no longer hold. $\mathcal{L}_{M,D,NO,U}$ can be validly encoded by $\mathcal{L}_{M,D,NM,U}$ via Theorem 3. The following proves the separation result.

Theorem 33. *There exists no valid encoding of $\mathcal{L}_{M,D,NM,U}$ into $\mathcal{L}_{M,D,NO,U}$.*

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

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

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

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

$\mathcal{L}_{M,D,NM,U}$ can be validly encoded into $\mathcal{L}_{M,C,NM,U}$ by Theorem 3, the reverse separation result holds by Theorem 1.

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

That $\mathcal{L}_{M,D,NM,U}$ is unrelated to $\mathcal{L}_{M,C,NO,U}$ follows from Theorems 33 & 1.

Theorem 37. *The languages $\mathcal{L}_{M,D,NM,U}$ and $\mathcal{L}_{M,C,NO,U}$ are unrelated.*

6.3 Equally Expressive Unification Languages

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

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. The next two language equivalences formalise this.

There is a trivial valid encoding of $\mathcal{L}_{P,D,NO,U}$ into $\mathcal{L}_{P,C,NO,U}$ by sub-language inclusion. The following shows equivalence via the reverse encoding $\mathcal{L}_{P,C,NO,U}$ into $\mathcal{L}_{P,D,NO,U}$.

Consider the encoding $[\![\cdot]\!]$ from $\mathcal{L}_{P,C,NO,U}$ to $\mathcal{L}_{P,D,NO,U}$ 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 38. *The encoding from $\mathcal{L}_{P,C,NO,U}$ into $\mathcal{L}_{P,D,NO,U}$ is valid.*

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

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

Theorem 39. *The languages $\mathcal{L}_{P,D,NM,U}$ and $\mathcal{L}_{P,C,NM,U}$ are equally expressive.*

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

Theorem 40. *There exist encodings from $\mathcal{L}_{P,-,I,U}$ into $\mathcal{L}_{M,-,I,U}$.*

Theorem 41. *There exist encodings from $\mathcal{L}_{-,C,I,U}$ into $\mathcal{L}_{-,D,I,U}$.*

6.4 Encodings into Polyadic Non-Intensional Languages

This section considers encodings into polyadic non-intensional unification languages. Despite being nominally no-matching it is still possible to encode polyadic name-matching into the languages $\mathcal{L}_{P,-,NO,U}$. 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.

Unification 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 exchange primitives, and so invalidates many prior results.

The directionality of asymmetric or exchange 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. Consider the encoding $[\![\cdot]\!]$ from $\mathcal{L}_{P,D,NM,E}$ to $\mathcal{L}_{P,C,NO,U}$ that exploits the functions $\textsc{Bin}$ and $\textsc{Not}$ of Section 5.2 and is homeomorphic on all forms except as defined below:

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

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. The unprotect function g converts name-matches into names since the former aren't defined in a no-matching language.

Theorem 42. *The encoding from $\mathcal{L}_{P,D,NM,E}$ into $\mathcal{L}_{P,C,NO,U}$ is valid.*

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

Relating to other unification languages, clearly $\mathcal{L}_{P,-,NO,U}$ can be validly encoded by $\mathcal{L}_{P,-,NM,U}$, with shifts between dataspace-based and channel-based communication handled by Theorems 38, 39 & 3. The following proves the separation result required to indicate an increase in expressiveness from $\mathcal{L}_{P,-,NO,U}$ to $\mathcal{L}_{P,-,NM,U}$.

Theorem 43. *There exists no valid encoding of $\mathcal{L}_{P,-,NM,U}$ into $\mathcal{L}_{P,-,NO,U}$.*

The relations between polyadic and monadic languages are as expected. $\mathcal{L}_{M,C,NO,U}$ can be encoded by $\mathcal{L}_{P,C,NO,U}$ via Theorem 3, and thus also $\mathcal{L}_{P,D,NO,U}$ by Theorem 38. The following separation result that $\mathcal{L}_{M,C,NO,U}$ cannot encode $\mathcal{L}_{P,-,NO,U}$ is by Theorem 1.

Theorem 44. *There exists no valid encoding of $\mathcal{L}_{P,-,NO,U}$ into $\mathcal{L}_{M,C,NO,U}$.*

Similar results hold for the name-matching languages also. $\mathcal{L}_{M,C,NM,U}$ can be encoded by $\mathcal{L}_{P,C,NM,U}$ by Theorem 3 (and thus also $\mathcal{L}_{P,D,NM,U}$ by Theorem 39). The reverse separation result that is proven via Theorem 1.

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

6.5 Intensional Unification Languages

Since all the intensional unification languages are equivalent by Theorems 40 & 41, it remains to show their other relations.

The intensional unification languages can also encode directionality in a similar manner to Section 6.4 (Theorem 42). Consider the encoding $[\![\cdot]\!]$ from $\mathcal{L}_{P,D,I,E}$ to $\mathcal{L}_{M,C,I,U}$ that exploits the numerical encoding function h and Nor of Sections 5.2 & 6.4 and is the homeomorphic on all forms except the action and co-action:

$$[\![(p_1, \ldots, p_i).P]\!] \stackrel{\text{def}}{=} a(\lambda \mathsf{rn} \bullet (p_1 \bullet \ldots \bullet p_i)).[\![P]\!] \qquad a = h(\widetilde{p})$$
$$[\![\langle p_1, \ldots, p_i \rangle.P]\!] \stackrel{\text{def}}{=} a(\mathsf{rn} \bullet (p_1 \bullet \ldots \bullet p_i)).[\![P]\!] \qquad a = \mathrm{No}\imath(h(\widetilde{p}))$$

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

Theorem 46. *The encoding from $\mathcal{L}_{P,D,I,E}$ into $\mathcal{L}_{M,C,I,U}$ is valid.*

That $\mathcal{L}_{-,-,I,U}$ are more expressive than $\mathcal{L}_{P,-,I,E}$ follows from the next theorem.

Theorem 47. *There exist valid encodings from $\mathcal{L}_{P,-,I,E}$ into $\mathcal{L}_{-,-,I,U}$.*

Finally, within the unification languages intensionality remains more expressive than non-intensionality. By Theorems 39 & 40 in all languages considered here being channel-based or dataspace-based is immaterial. The languages $\mathcal{L}_{P,-,NM,U}$ can be encoded by $\mathcal{L}_{P,-,I,U}$ by Theorem 3, the following separation completes the results.

Theorem 48. *There exist no valid encodings of $\mathcal{L}_{P,-,I,U}$ into $\mathcal{L}_{P,-,NM,U}$.*

7 Conclusions

Symmetric communication primitives provide new and interesting perspectives on how languages and communication can occur. Considering exchange provides interesting insight into how much trading systems and atomic exchange actions can be captured within asymmetric languages by encoding. The ability to encode polyadic exchange without name matching into asymmetric languages indicates that it is the addition of (unbounded, or multiple) name-matching with exchange that really extends expressiveness. While exchange generally increases expressiveness over asymmetric languages, it is pattern-matching that provides the strongest expressiveness alone. The unification languages cannot be encoded into even exchange languages, the self recognising technique was used for CPC before [11] but is here generalised. The flexibility of unification allows for names to be matched even in a language that nominally does not have name-matching. This yields some interesting results where non-matching languages can encode name-matching languages by exploiting unification. However, name-matching still provides increased expressiveness within unification languages, and intensionality is the sole factor in determining the most expressive language.

References

1. J. Bengtson, M. Johansson, J. Parrow, and B. Victor. Psi-calculi: a framework for mobile processes with nominal data and logic. *Logical Methods in Computer Science*, 7(1), 2011.
2. J. Bengtson and J. Parrow. Formalising the pi-calculus using nominal logic. *Logical Methods in Computer Science*, 5(2), 2009.
3. N. Busi, R. Gorrieri, and G. Zavattaro. On the expressiveness of linda coordination primitives. *Information and Computation*, 156(1-2):90–121, 2000.
4. R. De Nicola, D. Gorla, and R. Pugliese. On the expressive power of klaim-based calculi. *Theoretical Computer Science*, 356(3):387–421, May 2006.
5. D. Gelernter. Generative communication in Linda. *ACM Transactions on Programming Languages and Systems*, 7(1):80–112, 1985.
6. T. Given-Wilson. *Concurrent Pattern Unification*. PhD thesis, University of Technology, Sydney, Australia, 2012.
7. T. Given-Wilson. An intensional concurrent faithful encoding of turing machines. In *Proceedings ICE 2014, Berlin, Germany, 6th June 2014.*, pages 21–37, 2014.
8. T. Given-Wilson. On the Expressiveness of Intensional Communication. In *Proceedings of EXPRESS/SOS*, Rome, Italie, Sept. 2014.
9. T. Given-Wilson and D. Gorla. Pattern matching and bisimulation. In *Coordination Models and Languages*, volume 7890 of *Lecture Notes in Computer Science*, pages 60–74. Springer Berlin Heidelberg, 2013.
10. T. Given-Wilson, D. Gorla, and B. Jay. Concurrent pattern calculus. In *Theoretical Computer Science*, volume 323 of *IFIP Advances in Information and Communication Technology*, pages 244–258. Springer Berlin Heidelberg, 2010.
11. T. Given-Wilson, D. Gorla, and B. Jay. A Concurrent Pattern Calculus. *Logical Methods in Computer Science*, 10(3), 2014.
12. D. Gorla. Comparing communication primitives via their relative expressive power. *Information and Computation*, 206(8):931–952, 2008.
13. D. Gorla. A taxonomy of process calculi for distribution and mobility. *Distributed Computing*, 23(4):273–299, 2010.
14. D. Gorla. Towards a unified approach to encodability and separation results for process calculi. *Information and Computation*, 208(9):1031–1053, 2010.
15. K. Honda and N. Yoshida. On reduction-based process semantics. *Theoretical Computer Science*, 152:437–486, 1995.
16. R. Milner. The polyadic π-calculus: A tutorial. In *Logic and Algebra of Specification*, volume 94 of *Series F*. NATO ASI, Springer, 1993.
17. R. Milner, J. Parrow, and D. Walker. A calculus of mobile processes, I. *Information and Computation*, 100(1):1–40, Sept. 1992.
18. R. Milner, J. Parrow, and D. Walker. A calculus of mobile processes, II. *Information and Computation*, 100(1):41–77, Sept. 1992.
19. C. Palamidessi. Comparing the expressive power of the synchronous and asynchronous pi-calculi. *Mathematical. Structures in Comp. Sci.*, 13(5):685–719, Oct. 2003.
20. J. Parrow and B. Victor. The fusion calculus: expressiveness and symmetry in mobile processes. In *Proceedings of 13th Annual IEEE Symposium on Logic in Computer Science*, pages 176–185, Jun 1998.
21. C. Urban, S. Berghofer, and M. Norrish. Barendregts variable convention in rule inductions. In *Automated Deduction CADE-21*, volume 4603 of *Lecture Notes in Computer Science*, pages 35–50. Springer Berlin Heidelberg, 2007.

The appendices contain the following: Appendix A presents further diagrams of the results, indicating those within each language group. Appendix B discusses related work and some of the choices made in this paper that could be fit within the body of the text. Appendix C includes omitted results and proofs. Extended references conclude.

A Figures of Results

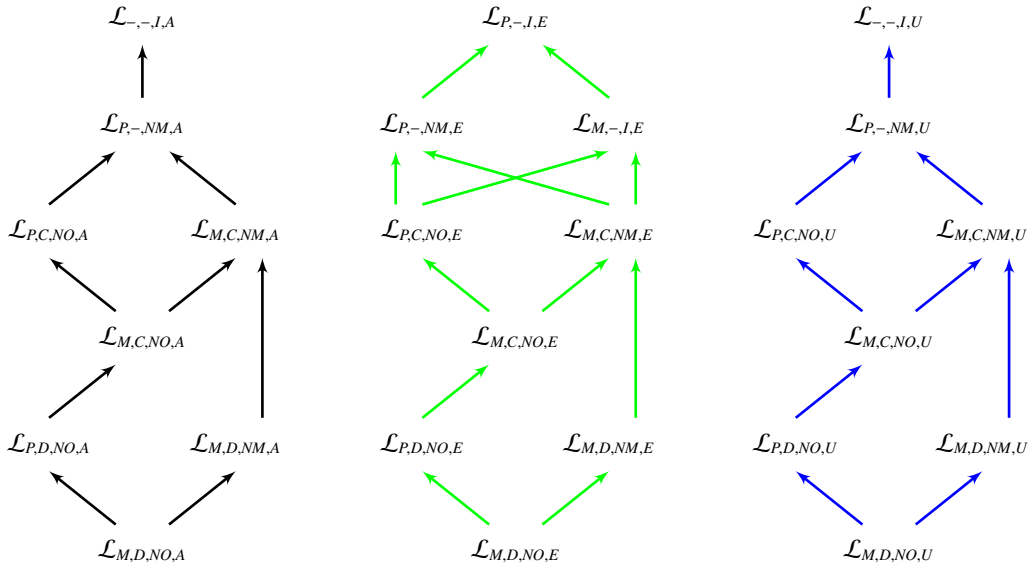

Fig. 3. Relations within Each Symmetric Language Groups

Figure 3 shows the relations within: the asymmetric languages as known from prior works; the exchange (green) languages as formalised in Section 5; and unification (blue) languages as formalised in Section 6. Arrows show increased expressive power and equality symbols show equal expressive power. Black arrows and equalities are from prior work, green from Section 5, and blue from Section 6. The lack of an arrow indicates no possible encoding in either direction (e.g. between $\mathcal{L}_{P,C,NO,E}$ and $\mathcal{L}_{M,C,NM,E}$). Transitive relations over greater distances where another language fits in between are omitted (e.g. $\mathcal{L}_{P,D,NO,E}$ to $\mathcal{L}_{P,C,NO,E}$). Note that as mentioned in Section 3, such transitive relations are ensured here by the encodings presented here, although they do not hold in general for valid encodings. Within the exchange languages only, this indicates the similarity to the asymmetric languages at the bottom of the diagram, with differences appearing at the top with polyadicity and intensionality having different expressiveness relations under exchange. The relations within the unification language group clearly mirror those of the asymmetric languages.

The combined diagram illustrating the results all together can be seen in Figure 2 in Section 4.

B Discussion

This appendix 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.

B.1 Choices of Primitives

The choices of primitives for the languages here is to align with prior work and also to exploit prior results [12, 14, 8]. 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 [6, 9]. It turns out that the (symmetric) patterns here are sufficient to represent most other approaches, such as Spi Calculus [11] and Psi Calculi terms [8]. 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 [30, 31, 6] and in many works related to *pattern calculus* and CPC.

For the process forms the most obvious alternative would be to consider a 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 [12, 14, 8]. The addition of such a choice operator could invalidate some findings, in particular Theorem 5.8 from [14] and Lemma 19 (that is key to Theorem 30). 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 [11], however alternative proofs also exist such as Theorems 1, 29, & 33 and in prior works [12, 8]. 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 for the encodings here, often more distant relations can be proved without relying on particular choices of primitives or proof techniques.

B.2 Related Work

This appendix 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 [19, 3, 24, 4, 29, 12, 10, 6, 11]. The choice of valid encodings here is to align with prior works [12, 14, 8] and where possible reuse prior results. These valid encodings are those used,

sometimes with mild adaptations, in [14, 13, 10, 35, 6, 11] and have also inspired similar works [32, 33, 38]. However, there are alternative approaches to encoding criteria or comparing expressive power [23, 27, 24, 36, 38]. Further arguments in favour of, or against, the valid encodings here can be found in [14, 13, 37, 38, 11].

Many of the languages here correspond to published languages. Observe that $\mathcal{L}_{M,C,NO,A}$, $\mathcal{L}_{P,C,NO,A}$, $\mathcal{L}_{M,C,NO,A}$, and $\mathcal{L}_{P,C,NO,A}$ use the communication paradigm of the asynchronous/synchronous monadic/polyadic π-calculus [17, 18, 16]. The language $\mathcal{L}_{P,D,NM,A}$ uses the communication paradigm of LINDA[5]; the languages $\mathcal{L}_{M,D,NO,A}$ and $\mathcal{L}_{P,D,NO,A}$ the communication paradigm of the monadic/polyadic Mobile Ambients [25]; and $\mathcal{L}_{P,C,NM,A}$ that of μKLAIM [34] or semantic-π [26]. Due to the large number of intensional languages here, many do not match the communication paradigm of well-known calculi. However, the language $\mathcal{L}_{M,D,I,A}$ is the asymmetric concurrent pattern calculus of [7] and calculi with other communication paradigms that match some of those here have been mentioned in [6], as variations of CPC [10, 6]. Similarly, the language $\mathcal{L}_{M,C,I,A}$ uses the communication paradigm of Spi calculus [22, 28] and Psi calculi (albeit with channel equivalence represented by equality and without the possibility of repeated binding names in patterns) [1]. There are also similarities between the communication paradigm of $\mathcal{L}_{M,C,I,A}$ and the polyadic synchronous π-calculus [24], although the intensionality in polyadic synchronous π-calculus is limited to the channel, i.e. inputs and outputs of the form $s(x).P$ and $\overline{s}\langle a\rangle.P$ respectively. With respect to non-asymmetric languages: $\mathcal{L}_{P,C,NO,E}$ is closest to the fusion calculus [20] although the scope of binding in communication is different; $\mathcal{L}_{M,D,I,U}$ corresponds to CPC; and $\mathcal{L}_{M,D,I,E}$, $\mathcal{L}_{M,C,I,E}$, and $\mathcal{L}_{M,C,I,U}$ to variants of CPC [6].

There are already existing specific results for some symmetric process calculi that agree with the results here. CPC ($\mathcal{L}_{M,D,I,U}$) 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 [10, 11, 8]. Meanwhile fusion calculus ($\mathcal{L}_{P,C,NO,E}$) 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 [10, 6, 11]. Similarly fusion calculus can encode π-calculi, although not the other way around [20]. Impossibility of encoding results for CPC and fusion calculus into many calculi can be derived from the results here.

C Omitted Proofs

This appendix contains results and proofs omitted from the main paper.

C.1 Proofs From Section 3

Proof (Proof of Theorem 3). The encoding is trivial and $[\![P]\!] \stackrel{\text{def}}{=} P$ for all forms. The proof is then straightforward and ensured by definition of the poly-match rule for the base reduction. For a detailed example of the proof technique see Theorem 8.

C.2 Proofs From Section 5

Proof (Proof of Theorem 4). The proof is by contradiction. Assume there exists a valid encoding $[\![\cdot]\!]$ from $\mathcal{L}_{M,D,NO,E}$ into $\mathcal{L}_{M,D,NO,A}$. Now consider the processes $P_1 = (\lambda x).P_1'$ and $P_2 = \langle a \rangle$. Since $P_1 \mid P_2 \longmapsto$ by validity of the encoding $[\![P_1 \mid P_2]\!] \longmapsto$ and this must be between some $R_1 = (\lambda z).R_1'$ and R_2. (This can be obtained by induction over the derivation tree for $[\![P_1 \mid P_2]\!] \longmapsto R$.) Observe that $R_1 \mid R_2$ cannot be a reduct of either $[\![P_1]\!]$ or $[\![P_2]\!]$ since then by Proposition 1 either P_1 or P_2 would reduce and this is not the case. Further, R_1 must be top level in $[\![P_i]\!]$ for $i \in \{1, 2\}$ by Proposition 2.

Now consider the processes $P_3 = \langle \lambda x \rangle.P_3'$ and $P_4 = (a)$. Since $P_3 \mid P_4 \longmapsto$ by validity of the encoding $[\![P_3 \mid P_4]\!] \longmapsto$ and this must be between some $R_3 = \langle c \rangle$ and R_4. Observe that $R_3 \mid R_4$ cannot be a reduct of either $[\![P_3]\!]$ or $[\![P_4]\!]$ since then by Proposition 1 either P_3 or P_4 would reduce and this is not the case. Thus, R_3 must be top level in $[\![P_j]\!]$ for $j \in \{3, 4\}$ by Proposition 2. Conclude by showing that $C_{|}^{\mathcal{N}}([\![P_i]\!], [\![P_j]\!]) \longmapsto$ in contradiction with Proposition 1.

Proof (Proof of Theorem 5). The proof technique is similar to Theorem 4. The main difference is to consider P_1' (and P_3') as having the form **if** $a = x$ **then** Q to ensure that there must be a binding name in the input of the form (λz) as otherwise P_1 (and P_3) would always yield Q or $\mathbf{0}$ (which can be shown to yield contradiction when $Q = \surd$).

Proof (Proof of Theorem 6). The proof is by contradiction. Assume there exists a valid encoding $[\![\cdot]\!]$ from $\mathcal{L}_{M,C,NM,E}$ into $\mathcal{L}_{M,C,NM,A}$. Consider the processes $P_1 = n(\lambda x).$**if** $x = b$ **then** $\surd$ and $P_2 = \overline{n}\langle a \rangle.P_2'$ where P_2' does not succeed. Since $P_1 \mid P_2 \longmapsto$ by validity of the encoding $[\![P_1 \mid P_2]\!] \longmapsto$ and this must be between some $R_1 = \overline{m_1}\langle c_1 \rangle.R_1'$ and R_2 for some m_1 and c_1 and R_1' and R_2. (This can be obtained by induction over the derivation tree for $[\![P_1 \mid P_2]\!] \longmapsto R$.) Observe that $R_1 \mid R_2$ cannot be a reduct of either $[\![P_1]\!]$ or $[\![P_2]\!]$ since then by Proposition 1 either P_1 or P_2 would reduce and this is not the case.

If R_1 is a top-level component of $[\![P_1]\!]$ then R_2 must be top-level in $[\![P_2]\!]$. Now consider R_2.

1. If R_2 is of the form $m_1(\lambda z).R_2'$ for some z and R_2' then consider the process $Q = n(\ulcorner a \urcorner).Q'$. Clearly $P_2 \mid Q \longmapsto$ and so $[\![P_2 \mid Q]\!] \longmapsto$ by Proposition 3. If this is from some S that is top level in $[\![Q]\!]$ then consider the substitution $\sigma = \{a/b\}$. Since $P_2 \mid \sigma Q \not\longmapsto$ then $R_2 \mid \sigma' S \not\longmapsto$ by Proposition 1, where σ' is defined by $[\![\sigma Q]\!] \cong \sigma' [\![Q]\!]$ and name invariance of $[\![\cdot]\!]$. If the $R_2 \mid \sigma' S \not\longmapsto$ then this can

only be due to a renaming of m_1 in $[\![Q]\!]$ by the poly-match rule. However, it can be shown that this renaming must also apply in $\sigma' R_1$ and so $\sigma' R_1 \mid R_2 \not\longmapsto$ yielding contradiction since $\sigma P_1 \mid P_2 \longmapsto$.

2. If R_2 is of the form $m_1(\ulcorner c_1 \urcorner).R'_2$ then consider the process $Q_1 = \overline{n}\langle b \rangle.Q'_1$. Clearly $P_1 \mid Q_1 \longmapsto$ and $P_2 \mid Q_1 \Downarrow$ and so $[\![P_1 \mid Q_1]\!]$ must also by Proposition 3 and success sensitiveness. By reasoning as in the above case and exploiting substitutions such as σ it follows that c_1 must not depend upon a or b since this would contradict operational correspondence, name invariance, or Propositions 1 or 3. Consider $Q_2 = n(\ulcorner a \urcorner).Q'_2$, clearly $P_2 \mid Q_2 \longmapsto$ and so $[\![P_2 \mid Q_2]\!] \longmapsto$ by Proposition 3 (and the substitution $\rho = \{b/a\}$). If $[\![Q_2]\!]$ interacts with R_2 then this must not depend upon a and so $[\![\rho Q_2 \mid P_2]\!] \longmapsto$ which contradicts Proposition 1 since $\rho Q_2 \mid P_2 \not\longmapsto$. Otherwise if $[\![Q_2]\!]$ interacts with some other top-level S in $[\![P_2]\!]$ then this can be shown to yield divergence as in Theorem 7.1 (sub-case 2) of [8].

If R_1 is a top-level component of $[\![P_2]\!]$ then R_2 must arise from $[\![P_1]\!]$. If R_2 is of the form $m_1(\ulcorner c_1 \urcorner).R'_2$ then the same approach as case 2 above can be used to show contradiction. Therefore R_2 must be of the form $m_1(\lambda z).R'_2$ for some z and R'_2. Now consider the process $Q = n(\ulcorner a \urcorner).Q'$. Since $Q \mid P_2 \longmapsto$ it follows that $[\![Q \mid P_2]\!] \longmapsto$ by Proposition 3. If the reduction $[\![Q \mid P_2]\!] \longmapsto$ does not involve R_1 then obtain contradiction via divergence as in Theorem 7.1 (sub-case 2) of [8]. Therefore, $[\![Q]\!]$ must include some top-level input $m_1(p)$ such that $\textsc{Match}(p, c_1)$ is defined. Now consider p. If p is of the form λw then the reduction $[\![Q \mid P_2]\!] \longmapsto$ cannot depend upon the name a in Q and contradiction is yielded as in case 1 above. Finally, if p is of the form $\ulcorner c_1 \urcorner$ then $m_1(\ulcorner c_1 \urcorner)$ cannot bind any names, and so there cannot be a restricted channel used for further interaction as in the encoding from $\mathcal{L}_{M,D,NO,E}$ into $\mathcal{L}_{M,C,NO,A}$. This can be exploited to show that since there is no restricted channel, the processes $P_3 = \overline{n}\langle \lambda x \rangle.\textbf{if } x = c \textbf{ then } \Omega$ and $P_4 = n(b).P'_4$ (where Ω is a divergent process) can yield contradiction by one of: $[\![P_1 \mid P_4]\!] \longmapsto$ when $P_1 \mid P_4 \not\longmapsto$, or $[\![P_3 \mid P_2]\!] \longmapsto$ when $P_3 \mid P_2 \not\longmapsto$, or choosing ρ such that $[\![\rho(P_1 \mid P_2 \mid P_3 \mid P_4)]\!]$ both succeeds and diverges, when $\rho(P_1 \mid P_2 \mid P_3 \mid P_4)$ can only succeed or diverge but not both.

Theorem 49. *There exists no valid encoding of $\mathcal{L}_{M,C,NO,E}$ into $\mathcal{L}_{M,D,NO,E}$.*

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

Theorem 50. *There exists no valid encoding of $\mathcal{L}_{M,D,NM,E}$ into $\mathcal{L}_{M,D,NO,E}$.*

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

Theorem 51. *There exists no valid encoding of $\mathcal{L}_{M,C,NM,E}$ into $\mathcal{L}_{M,D,NM,E}$.*

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

Theorem 52. *There exists no valid encoding of $\mathcal{L}_{M,C,NM,E}$ into $\mathcal{L}_{M,C,NO,E}$.*

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

Proof (Proof of Theorem 7). To show there is no valid encoding from $\mathcal{L}_{M,D,NM,E}$ into $\mathcal{L}_{M,C,NO,E}$ 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 1.

In the other direction the technique of Theorem 4.5 of [12] can be used.

The following lemmas are used in the proof below.

Lemma 1. *If $P \equiv Q$ then $[\![P]\!] \equiv [\![Q]\!]$. Conversely, if $[\![P]\!] \equiv Q$ then $Q = [\![P']\!]$ for some $P' \equiv P$.*

Proof. Straightforward, the only non-trivial part is when renaming has occurred.

Lemma 2. *Given $\mathcal{L}_{M,C,NO,E}$ action P and co-action Q then $[\![P]\!] \mid [\![Q]\!] \longmapsto$ if and only if $P \mid Q \longmapsto$.*

Proof. Both parts can be proved by induction on the height of the proof tree for the judgments $[\![P \mid Q]\!] \longmapsto$ and $P \mid Q \longmapsto$. The base case is ensured by the poly-match rule when P is of the form $n(\lambda x).P'$ or $\overline{n}\langle \lambda x \rangle.P'$. Note that Lemma 1 is used to ensure structural congruence.

Lemma 3. *The translation $[\![\cdot]\!]$ from $\mathcal{L}_{M,D,NO,E}$ 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 judgments $P \longmapsto P'$ and $[\![P]\!] \longmapsto Q$, respectively. In both cases, the base step is the most interesting and follows from Lemma 2, 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 1.

Proof (Proof of Theorem 8). Compositionality and name invariance hold by construction. Operational correspondence (with structural equivalence in the place of $\cong$) and divergence reflection follow from Lemma 3. 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 3 k times and Lemma 1 obtain that $[\![P]\!] \longmapsto^{3k} [\![P']\!] \equiv [\![P'']\!] \mid \surd$, i.e. that $[\![P]\!] \Downarrow$. The converse implication can be proved similarly.

The following lemmas are used in the proof below.

Lemma 4. *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 5. *Given $\mathcal{L}_{M,D,NO,E}$ action P and co-action Q then $[\![P]\!] \mid [\![Q]\!] \longmapsto$ if and only if $P \mid Q \longmapsto$.*

Proof. Both parts can be proved by induction on the height of the proof tree for the judgments $[\![P \mid Q]\!] \longmapsto$ and $P \mid Q \longmapsto$. The base case is ensured by the poly-match rule when P is of the form $(\lambda x).P'$ or $\langle \lambda x \rangle.P'$. Note that Lemma 4 is used to ensure structural congruence.

Lemma 6. *The translation $[\![\cdot]\!]$ from $\mathcal{L}_{M,D,NO,E}$ 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 judgments $P \longmapsto P'$ and $[\![P]\!] \longmapsto Q$, respectively. In both cases, the base step is the most interesting and follows from Lemma 5, 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 4.

Proof (Proof of Theorem 9). Compositionality and name invariance hold by construction. Operational correspondence (with structural equivalence in the place of $\cong$) and divergence reflection follow from Lemma 6. 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 6 k times and Lemma 4 obtain that $[\![P]\!] \longmapsto^k [\![P']\!] \equiv [\![P'']\!] \mid \surd$, i.e. that $[\![P]\!] \Downarrow$. The converse implication can be proved similarly.

Proof (Proof of Theorem 10). That $\mathcal{L}_{M,C,NO,A}$ cannot be encoded by $\mathcal{L}_{M,D,NO,E}$ follows from Theorem 1 since the matching degree of $\mathcal{L}_{M,C,NO,A}$ is 1 and the matching degree of $\mathcal{L}_{M,D,NO,E}$ is 0.

The following lemmas are used in the proof of the theorem below.

Lemma 7. *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 8. *Given $\mathcal{L}_{M,D,NM,E}$ action P and co-action Q then $[\![P]\!] \mid [\![Q]\!] \longmapsto$ if and only if $P \mid Q \longmapsto$.*

Proof. Both parts can be proved by induction on the height of the proof tree for the judgments $[\![P \mid Q]\!] \longmapsto$ and $P \mid Q \longmapsto$. The base case is ensured by the poly-match rule when P is of the form $(\lambda x).P'$ or $\langle \lambda x \rangle.P'$ or $(\ulcorner a \urcorner).P'$ or $\langle \ulcorner a \urcorner \rangle.P'$. Note that Lemma 7 is used to ensure structural congruence.

Lemma 9. *The translation* $[\![\cdot]\!]$ *from* $\mathcal{L}_{M,D,NM,E}$ *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 judgments $P \longmapsto P'$ and $[\![P]\!] \longmapsto Q$, respectively. In both cases, the base step is the most interesting and follows from Lemma 8, 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 7.

Proof (Proof of Theorem 11). Compositionality and name invariance hold by construction. Operational correspondence (with structural equivalence in the place of $\cong$) and divergence reflection follow from Lemma 9. 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 9 k times and Lemma 7 obtain that $[\![P]\!] \longmapsto^k [\![P']\!] \equiv [\![P'']\!] \mid \surd$, i.e. that $[\![P]\!] \Downarrow$. The converse implication can be proved similarly.

Proof (Proof of Theorem 12). That $\mathcal{L}_{M,C,NM,A}$ cannot be encoded by $\mathcal{L}_{M,D,NM,E}$ follows from Theorem 1 since the matching degree of $\mathcal{L}_{M,C,NO,A}$ is 2 and the matching degree of $\mathcal{L}_{M,D,NO,E}$ is 1.

Proof (Proof of Theorem 13). The proof is a straightforward adaption the proof of Theorem 11 by taking the translation $[\![\cdot]\!]$ that is homomorphic on all forms except:

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

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

Proof (Proof of Theorem 14). By Theorem 1 since the matching degree of $\mathcal{L}_{P,-,NM,A}$ is ∞ and the matching degree of $\mathcal{L}_{M,C,NM,E}$ is 2.

The following lemmas are used in the proof below.

Lemma 10. *If* $P \equiv Q$ *then* $[\![P]\!] \equiv [\![Q]\!]$*. Conversely, if* $[\![P]\!] \equiv Q$ *then* $Q = [\![P']\!]$ *for some* $P' \equiv P$*.*

Proof. Straightforward, the only non-trivial part is when there is $(\nu \mathsf{rn})$ from the encoding of a co-action.

Lemma 11. *Given* $\mathcal{L}_{P,D,NO,E}$ *action* P *and co-action* Q *then* $[\![P]\!] \mid [\![Q]\!] \longmapsto$ *if and only if* $P \mid Q \longmapsto$*.*

Proof. Both parts can be proved by induction on the height of the proof tree for the judgments $[\![P \mid Q]\!] \longmapsto$ and $P \mid Q \longmapsto$. The base case is when P is of the form $(\widetilde{p}).P'$ and is ensured induction on the arity of the input and by the poly-match rule. Note that Lemma 10 is used to ensure structural congruence.

Lemma 12. *The translation* $[\![\cdot]\!]$ *from* $\mathcal{L}_{P,D,NO,E}$ *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 judgments $P \longmapsto P'$ and $[\![P]\!] \longmapsto Q$, respectively. In both cases, the base step is the most interesting and follows from Lemma 11, 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 10.

Proof (Proof of Theorem 15). Compositionality and name invariance hold by construction. Operational correspondence (with structural equivalence in the place of $\cong$) and divergence reflection follow from Lemma 12. 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 12 k times and Lemma 10 obtain that $[\![P]\!] \longmapsto^k [\![P']\!] \equiv [\![P'']\!] \mid \surd$, i.e. that $[\![P]\!] \Downarrow$. The converse implication can be proved similarly.

Proof (Proof of Theorem 16). By Theorem 1 since the matching degree of $\mathcal{L}_{P,C,NO,A}$ is 1 and the matching degree of $\mathcal{L}_{P,D,NO,E}$ is 0.

Proof (Proof of Theorem 17). The same as Theorem 15.

Proof (Proof of Theorem 18). The proof is by contradiction. Assume there exists a valid encoding $[\![\cdot]\!]$ from $\mathcal{L}_{P,D,NO,E}$ into $\mathcal{L}_{M,D,NO,E}$. 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 $[\![P_1 \mid P_2 \mid P_3 \mid P_4]\!] \mapsto\!\!\!\!\!\!\rightarrow$ or $[\![P_1 \mid P_2 \mid P_3 \mid P_4]\!] \Downarrow$, both of which yield contradiction.

Proof (Proof of Theorem 19). The same technique as Theorem 18 can be used by having all communication along a single channel name and preventing modification of this name by the encoding.

Proof (Proof of Theorem 20). The proof is by contradiction. Assume there is a valid encoding of $\mathcal{L}_{P,D,NO,E}$ into $\mathcal{L}_{P,D,NO,A}$. The key to the proof is to consider the processes $P_1 = \mathbf{if}\ x = a\ \mathbf{then}\ (\lambda z).\surd$ and $P_2 = \mathbf{if}\ y = d\ \mathbf{then}\ \langle z \rangle.\mathbf{0}$. 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.

The following lemmas are used in the proof of the theorem below.

Lemma 13. *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 14. *Given $\mathcal{L}_{M,C,I,E}$ action P and co-action Q then $[\![P]\!] \mid [\![Q]\!] \longmapsto$ if and only if $P \mid Q \longmapsto$.*

Proof. Both parts can be proved by induction on the height of the proof tree for the judgments $[\![P \mid Q]\!] \longmapsto$ and $P \mid Q \longmapsto$. The base case is when P is of the form $p(q).P'$ or $\overline{p}\langle q \rangle.P'$ where q is an intensional pattern, this is ensured by the poly-match rule. Note that Lemma 13 is used to ensure structural congruence.

Lemma 15. *The translation $[\![\cdot]\!]$ from $\mathcal{L}_{M,C,I,E}$ into $\mathcal{L}_{M,D,I,E}$ 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 judgments $P \longmapsto P'$ and $[\![P]\!] \longmapsto Q$, respectively. In both cases, the base step is the most interesting and follows from Lemma 14, 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 13.

Proof (Proof of Theorem 21). Compositionality and name invariance hold by construction. Operational correspondence (with structural equivalence in the place of $\cong$) and divergence reflection follow from Lemma 15. 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 15 k times and Lemma 13 obtain that $[\![P]\!] \longmapsto^k [\![P']\!] \equiv [\![P'']\!] \mid \surd$, i.e. that $[\![P]\!] \Downarrow$. The converse implication can be proved similarly.

Proof (Proof of Theorem 22). The proof is a straightforward adaption of the encoding and proof of Theorem 9.

The following lemmas are used in the theorem proof below.

Lemma 16. *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 17. *Given $\mathcal{L}_{P,C,NM,E}$ action P and co-action Q then $[\![P]\!] \mid [\![Q]\!] \longmapsto$ if and only if $P \mid Q \longmapsto$.*

Proof. Both parts can be proved by induction on the height of the proof tree for the judgments $[\![P \mid Q]\!] \longmapsto$ and $P \mid Q \longmapsto$. The base case is when P is of the form $a(\widetilde{p}).P'$ and is ensured induction on the arity of the input and by the poly-match rule. Note that Lemma 16 is used to ensure structural congruence.

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

Lemma 18. *The translation $[\![\cdot]\!]$ from $\mathcal{L}_{P,C,NM,E}$ into $\mathcal{L}_{P,D,NM,E}$ 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 17, 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 16.

Proof (Proof of Theorem 23). Compositionality and name invariance hold by construction. Operational correspondence (with structural equivalence in the place of $\cong$) and divergence reflection follow from Lemma 18. 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 18 k times and Lemma 16 obtain that $[\![P]\!] \longmapsto^k [\![P']\!] \equiv [\![P'']\!] \mid \surd$, i.e. that $[\![P]\!] \Downarrow$. The converse implication can be proved similarly.

Proof (Proof of Theorem 24). In the same manner as Theorem 23.

Proof (Proof of Theorem 25). The proof technique is the same as Theorem 7.1 of [8]. 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 results in divergence.

Proof (Proof of Theorem 26). The proof technique as Theorem 29 can be used here.

Proof (Proof of Theorem 27). The proof of Theorem 25 can be used here.

Proof (Proof of Theorem 28). The proof technique of Theorem 18 can be used here.

Proof (Proof of Theorem 29). 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 $\sigma'_1(\sigma'_2([\![P_1 \mid P_3]\!])) \mapsto\!\!\!\!\!\!\rightarrow$ or $\sigma'_1(\sigma'_2([\![P_1 \mid P_2 \mid P_3]\!]))$ is divergent, both of which yield contradiction.

C.3 Proofs From Section 6

The following lemma is used for the proof of the theorem below.

Lemma 19. *Given a language* $\mathcal{L}_1$ *that it non-unification, 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 $S \longmapsto S'$ this contradicts $S \not\longmapsto$.

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 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.

Proof (Proof of Theorem 30). Observe that in all the unification languages the self recognising process S is such that $S \not\longmapsto$ and $S \not\Downarrow$, however $S \mid S \longmapsto$ and $S \mid S \Downarrow$. Conclude by Theorem 2 and Lemma 19.

Theorem 53. *There exists no valid encoding from $\mathcal{L}_{M,D,NO,U}$ into $\mathcal{L}_{M,D,NO,\delta}$ where $\delta \neq U$.*

Proof. By Theorem 30.

Proof (Proof of Theorem 31). By contradiction. Assume there exists a valid encoding $[\![\cdot]\!]$ from $\mathcal{L}_{M,D,NO,\delta}$ where $\delta \neq U$ into $\mathcal{L}_{M,D,NO,U}$. 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 $\mathcal{C}_|^{\mathcal{N}}([\![P_1]\!], [\![P_2]\!])$ 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 $\mathcal{C}_|^{\mathcal{N}}([\![P_i]\!], [\![P_i]\!])$ can be shown to reduce (and report success), however this contradicts that $P_i \mid P_i \not\longmapsto$.

Theorem 54. *There exists no valid encoding from $\mathcal{L}_{M,\beta,\gamma,U}$ into $\mathcal{L}_{M,\beta,\gamma,\delta}$ where $\gamma \leq NM$ and $\delta \neq U$.*

Proof. By Theorem 30.

Proof (Proof of Theorem 32). The same technique as Theorem 31 can be used here.

Proof (Proof of Theorem 33). The proof is by contradiction, assume that there exists a valid encoding $[\![\cdot]\!]$ from $\mathcal{L}_{M,D,NM,U}$ into $\mathcal{L}_{M,D,NO,U}$. 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.

Proof (Proof of Theorem 34). The same technique as Theorem 33 works here.

Proof (Proof of Theorem 35). Observe that the matching degree of $\mathcal{L}_{M,C,NO,U}$ is 2 and the matching degree of $\mathcal{L}_{M,D,NO,U}$ is 1, conclude by Theorem 1.

Proof (Proof of Theorem 36). Observe that the matching degree of $\mathcal{L}_{M,C,NM,U}$ is 2 and the matching degree of $\mathcal{L}_{M,D,NM,U}$ is 1, conclude by Theorem 1.

Proof (Proof of Theorem 37). That there exists no valid encoding from $\mathcal{L}_{M,D,NM,U}$ into $\mathcal{L}_{M,C,NO,U}$ follows from Theorem 33. In the reverse direction the matching degree suffices to show separation.

The following lemmas are used in the proof of the theorem below.

Lemma 20. *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 21. *Given $\mathcal{L}_{P,C,NO,U}$ actions P and Q then $[\![P]\!] \mid [\![Q]\!] \longmapsto$ if and only if $P \mid Q \longmapsto$.*

Proof. Both parts can be proved by induction on the height of the proof tree for the judgments $[\![P \mid Q]\!] \longmapsto$ and $P \mid Q \longmapsto$. The base case is when P is of the form $a(\widetilde{p}).P'$ and is ensured induction on the arity of the input and by the poly-match rule. Note that Lemma 20 is used to ensure structural congruence.

Lemma 22. *The translation $[\![\cdot]\!]$ from $\mathcal{L}_{P,C,NO,U}$ into $\mathcal{L}_{P,D,NO,U}$ 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 judgments $P \longmapsto P'$ and $[\![P]\!] \longmapsto Q$, respectively. In both cases, the base step is the most interesting and follows from Lemma 21, 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 20.

Proof (Proof of Theorem 38). Compositionality and name invariance hold by construction. Operational correspondence (with structural equivalence in the place of $\cong$) and divergence reflection follow from Lemma 22. 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 22 k times and Lemma 20 obtain that $[\![P]\!] \longmapsto^k [\![P']\!] \equiv [\![P'']\!] \mid \surd$, i.e. that $[\![P]\!] \Downarrow$. The converse implication can be proved similarly.

Proof (Proof of Theorem 39). That $\mathcal{L}_{P,D,NM,U}$ can be validly encoded into $\mathcal{L}_{P,C,NM,U}$ follows by sub-language inclusion. The reverse direction can be shown in the same manner as for Theorem 38.

Proof (Proof of Theorem 40). The same technique as Theorem 5.4 of [8] 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 $(\mathsf{rn} \bullet p_1) \bullet \ldots \bullet p_i$ where rn is a reserved name.

Proof (Proof of Theorem 41). The same technique as used in Theorem 21 can be used here.

The following lemmas are used in the proof of the theorem below.

Lemma 23. *If $P \equiv Q$ then $[\![P]\!] \equiv [\![Q]\!]$. Conversely, if $[\![P]\!] \equiv Q$ then $Q = [\![P']\!]$ for some $P' \equiv P$.*

Proof. Straightforward, the only non-trivial part is when there is $(\nu \mathsf{rn})$ from the encoding of a co-action.

Lemma 24. *Given $\mathcal{L}_{P,D,NO,E}$ action P and co-action Q then $[\![P]\!] \mid [\![Q]\!] \longmapsto$ if and only if $P \mid Q \longmapsto$.*

Proof. Both parts can be proved by induction on the height of the proof tree for the judgments $[\![P \mid Q]\!] \longmapsto$ and $P \mid Q \longmapsto$. The base case is when P is of the form $(\widetilde{p}).P'$ and is ensured induction on the arity of the input and by the poly-match rule. Note that Lemma 23 is used to ensure structural congruence.

Lemma 25. *The translation $[\![\cdot]\!]$ from $\mathcal{L}_{P,D,NM,E}$ into $\mathcal{L}_{P,C,NO,U}$ 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 24, 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 23.

Proof (Proof of Theorem 42). Compositionality and name invariance hold by construction. Operational correspondence (with structural equivalence in the place of $\cong$) and divergence reflection follow from Lemma 25. 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 25 k times and Lemma 23 obtain that $[\![P]\!] \longmapsto^k [\![P']\!] \equiv [\![P'']\!] \mid \surd$, i.e. that $[\![P]\!] \Downarrow$. The converse implication can be proved similarly.

Proof (Proof of Theorem 43). The same technique as Theorem 33 can be used here to show no encoding from a name-matching unification language into a non-matching unification language.

Proof (Proof of Theorem 44). Observe that the matching degree of $\mathcal{L}_{P,-,NO,U}$ is ∞ and the matching degree of $\mathcal{L}_{M,C,NO,U}$ is 2, conclude by Theorem 1.

Proof (Proof of Theorem 45). Observe that the matching degree of $\mathcal{L}_{P,-,NM,U}$ is ∞ and the matching degree of $\mathcal{L}_{M,C,NM,U}$ is 2, conclude by Theorem 1.

The following lemmas are used in the proof of the theorem below.

Lemma 26. *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 27. *Given $\mathcal{L}_{P,D,I,E}$ action P and co-action Q then $[\![P]\!] \mid [\![Q]\!] \longmapsto$ if and only if $P \mid Q \longmapsto$.*

Proof. Both parts can be proved by induction on the height of the proof tree for the judgments $[\![P \mid Q]\!] \longmapsto$ and $P \mid Q \longmapsto$. The base case is when P is of the form $(\widetilde{p}).P'$ and is ensured induction on the arity of the input and by the poly-match rule. Note that Lemma 26 is used to ensure structural congruence.

Lemma 28. *The translation $[\![\cdot]\!]$ from $\mathcal{L}_{P,D,I,E}$ into $\mathcal{L}_{M,C,I,U}$ 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 27, 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 26.

Proof (Proof of Theorem 46). Compositionality and name invariance hold by construction. Operational correspondence (with structural equivalence in the place of $\cong$) and divergence reflection follow from Lemma 28. 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 28 k times and Lemma 26 obtain that $[\![P]\!] \longmapsto^k [\![P']\!] \equiv [\![P'']\!] \mid \surd$, i.e. that $[\![P]\!] \Downarrow$. The converse implication can be proved similarly.

Proof (Proof of Theorem 47). There exists an encoding from $\mathcal{L}_{P,D,I,E}$ into $\mathcal{L}_{M,C,I,U}$ by Theorem 46. The rest follow by the techniques used in Theorems 24, 40 & 41.

Theorem 55. *There exist no valid encodings from $\mathcal{L}_{-,-,I,U}$ into $\mathcal{L}_{P,-,I,E}$.*

Proof. For all combinations the proof follows from Theorem 30.

Proof (Proof of Theorem 48). The same technique as Theorem 25 applies here.

Extended References

1. J. Bengtson, M. Johansson, J. Parrow, and B. Victor. Psi-calculi: a framework for mobile processes with nominal data and logic. *Logical Methods in Computer Science*, 7(1), 2011.
2. J. Bengtson and J. Parrow. Formalising the pi-calculus using nominal logic. *Logical Methods in Computer Science*, 5(2), 2009.
3. N. Busi, R. Gorrieri, and G. Zavattaro. On the expressiveness of linda coordination primitives. *Information and Computation*, 156(1-2):90–121, 2000.
4. R. De Nicola, D. Gorla, and R. Pugliese. On the expressive power of klaim-based calculi. *Theoretical Computer Science*, 356(3):387–421, May 2006.
5. D. Gelernter. Generative communication in Linda. *ACM Transactions on Programming Languages and Systems*, 7(1):80–112, 1985.
6. T. Given-Wilson. *Concurrent Pattern Unification*. PhD thesis, University of Technology, Sydney, Australia, 2012.
7. T. Given-Wilson. An intensional concurrent faithful encoding of turing machines. In *Proceedings ICE 2014, Berlin, Germany, 6th June 2014.*, pages 21–37, 2014.
8. T. Given-Wilson. On the Expressiveness of Intensional Communication. In *Proceedings of EXPRESS/SOS*, Rome, Italie, Sept. 2014.
9. T. Given-Wilson and D. Gorla. Pattern matching and bisimulation. In *Coordination Models and Languages*, volume 7890 of *Lecture Notes in Computer Science*, pages 60–74. Springer Berlin Heidelberg, 2013.
10. T. Given-Wilson, D. Gorla, and B. Jay. Concurrent pattern calculus. In *Theoretical Computer Science*, volume 323 of *IFIP Advances in Information and Communication Technology*, pages 244–258. Springer Berlin Heidelberg, 2010.
11. T. Given-Wilson, D. Gorla, and B. Jay. A Concurrent Pattern Calculus. *Logical Methods in Computer Science*, 10(3), 2014.
12. D. Gorla. Comparing communication primitives via their relative expressive power. *Information and Computation*, 206(8):931–952, 2008.
13. D. Gorla. A taxonomy of process calculi for distribution and mobility. *Distributed Computing*, 23(4):273–299, 2010.
14. D. Gorla. Towards a unified approach to encodability and separation results for process calculi. *Information and Computation*, 208(9):1031–1053, 2010.
15. K. Honda and N. Yoshida. On reduction-based process semantics. *Theoretical Computer Science*, 152:437–486, 1995.
16. R. Milner. The polyadic π-calculus: A tutorial. In *Logic and Algebra of Specification*, volume 94 of *Series F*. NATO ASI, Springer, 1993.
17. R. Milner, J. Parrow, and D. Walker. A calculus of mobile processes, I. *Information and Computation*, 100(1):1–40, Sept. 1992.
18. R. Milner, J. Parrow, and D. Walker. A calculus of mobile processes, II. *Information and Computation*, 100(1):41–77, Sept. 1992.
19. C. Palamidessi. Comparing the expressive power of the synchronous and asynchronous pi-calculi. *Mathematical. Structures in Comp. Sci.*, 13(5):685–719, Oct. 2003.
20. J. Parrow and B. Victor. The fusion calculus: expressiveness and symmetry in mobile processes. In *Proceedings of 13th Annual IEEE Symposium on Logic in Computer Science*, pages 176–185, Jun 1998.
21. C. Urban, S. Berghofer, and M. Norrish. Barendregts variable convention in rule inductions. In *Automated Deduction CADE-21*, volume 4603 of *Lecture Notes in Computer Science*, pages 35–50. Springer Berlin Heidelberg, 2007.
22. M. Abadi and A. D. Gordon. A calculus for cryptographic protocols: The spi calculus. In *Proceedings of the 4th ACM Conference on Computer and Communications Security*, CCS '97, pages 36–47, New York, NY, USA, 1997. ACM.

23. G. Boudol. Notes on algebraic calculi of processes. In *Logics and Models of Concurrent Systems*, pages 261–303. Springer-Verlag, New York, NY, USA, 1985.
24. M. Carbone and S. Maffeis. On the expressive power of polyadic synchronisation in π-calculus. *Nordic Journal of Computing*, 10(2):70–98, May 2003.
25. L. Cardelli and A. D. Gordon. Mobile ambients. In *FoSSaCS '98*, pages 140–155, 1998.
26. G. Castagna, R. D. Nicola, and D. Varacca. Semantic subtyping for the pi-calculus. *Theoretical Computer Science*, 398(1-3):217–242, May 2008.
27. R. de Simone. Higher-level synchronising devices in Meije-SCCS. *Theoretical Computer Science*, 37:245–267, 1985.
28. C. Haack and A. Jeffrey. Pattern-matching spi-calculus. *Information and Computation*, 204(8):1195–1263, Aug. 2006.
29. B. Haagensen, S. Maffeis, and I. Phillips. Matching systems for concurrent calculi. *Electronic Notes in Theoretical Computer Science*, 194(2):85 – 99, 2008. Proceedings of EXPRESS.
30. B. Jay. *Pattern Calculus: Computing with Functions and Data Structures*. Springer, 2009.
31. B. Jay and T. Given-Wilson. A combinatory account of internal structure. *Journal of Symbolic Logic*, 76(3):807–826, 2011.
32. I. Lanese, J. Prez, D. Sangiorgi, and A. Schmitt. On the expressiveness of polyadic and synchronous communication in higher-order process calculi. In *Automata, Languages and Programming*, volume 6199 of *Lecture Notes in Computer Science*, pages 442–453. Springer Berlin Heidelberg, 2010.
33. I. Lanese, C. Vaz, and C. 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.
34. R. D. Nicola, G. L. Ferrari, and R. Pugliese. KLAIM: A kernel language for agents interaction and mobility. *IEEE Transactions on Software Engineering*, 24(5):315–330, 1998.
35. L. Nielsen, N. Yoshida, and K. Honda. Multiparty symmetric sum types. In *Proceedings of EXPRESS*, pages 121–135, 2010.
36. J. Parrow. Expressiveness of process algebras. *Electronic Notes in Theoretical Computer Science*, 209:173–186, Apr. 2008.
37. K. Peters. Translational expressiveness: comparing process calculi using encodings. 2012.
38. R. J. van Glabbeek. Musings on encodings and expressiveness. In *Proceedings of EXPRESS/SOS*, volume 89 of *EPTCS*, pages 81–98, 2012.