Matching in the pi-calculus

Kirstin Peters, Tsvetelina Yonova-Karbe, Uwe Nestmann

Angaben zur Veröffentlichung / Publication details:

Peters, Kirstin, Tsvetelina Yonova-Karbe, and Uwe Nestmann. 2014. "Matching in the pi-calculus." *Electronic Proceedings in Theoretical Computer Science* 160: 16–29. https://doi.org/10.4204/eptcs.160.3.

Matching in the Pi-Calculus

Kirstin Peters Tsvetelina Yonova-Karbe Uwe Nestmann
TU Berlin, Germany

We study whether, in the π-calculus, the match prefix—a conditional operator testing two names for (syntactic) equality—is expressible via the other operators. Previously, Carbone and Maffeis proved that matching is *not* expressible this way under rather strong requirements (preservation and reflection of observables). Later on, Gorla developed a by now widely-tested set of criteria for encodings that allows much more freedom (e.g. instead of direct translations of observables it allows comparison of calculi with respect to reachability of successful states). In this paper, we offer a considerably stronger separation result on the non-expressibility of matching using only Gorla's relaxed requirements.

1 Introduction

In process calculi matching is a simple mechanism to trigger a process if two names are syntactically equal. The match prefix $[a = b]P$ in the π-calculus works as a conditional guard. If the names a and b are identical the process behaves as P. Otherwise, the term cannot reduce further.

Motivation. The principle of matching two names in order to reduce a term is also present in another form in any calculus with channel-based synchronisation, like CCS or the π-calculus. The rule for communication demands identical (i.e. matching) input/output channel names to be used by parallel processes. For example, the term $\overline{a} \mid a.P$ may communicate on a, but the term $\overline{a} \mid b.P$ cannot communicate at all. Thus, the π-calculus already contains a "distributed" form of the match prefix.[1] However, it is also an "unprotected" and therefore non-deterministic form of matching, as $\overline{a} \mid a.P \mid a.Q$ allows for two different communications. This raises the natural question whether the match prefix can be encoded using the other operations of the calculus, or whether it is a basic construct. Here we show that communication is indeed the π-calculus construct that is closest to the match prefix. Accordingly an encoding of the match prefix would need to translate the prefix into a (set of) communication step(s) on links that result from the translation of the match variables. These links have to be free—to allow for a guarding input to receive a value for a match variable—but they also have to be bound—to avoid unintended interactions between parallel match encodings. This kind of binding cannot be simulated by a π-calculus operator different from the match prefix. Thus the match prefix is a basic construct of the π-calculus and cannot be encoded. Note that, as shown by the use of the match prefix e.g. in [9] for a sound axiomatisation of late congruence, in [18] for a complete axiomatisation of open equivalence, or—more recently—in [4] for a session pi-calculus, the match prefix is regarded as useful, i.e. it allows for applications that without the match prefix are not possible or more complicated to achieve. Thus a better understanding of the nature of the match prefix contributes to current research.

[1] Of course, this observation extends to Linda-like tuple-based communication, and even to Actor-like message routing according to the matching object identity.

Quality criteria. Of course, we are not interested in trivial or meaningless encodings. Instead we consider only those encodings that ensure that the original term and its encoding show to some extent the same abstract behaviour. To analyse the quality of encodings and to rule out trivial or meaningless encodings, they are evaluated w.r.t. a set of quality criteria. Note that stricter criteria that rule out more encoding attempts strengthen an *encodability result*, i.e. the proof of the existence of an encoding between two languages that respects the criteria. A stronger encodability result reveals a closer connection between the considered languages. In contrast weaker criteria strengthen a *separation result*, i.e. the proof of the non-existence of an encoding between two languages w.r.t. the criteria. A stronger separation result illuminates a conceptional difference between two languages, i.e. some kind of behaviour of the source language that cannot be simulated by the target language. Unfortunately there is no consensus about what properties make an encoding "good" or "good enough" to compare two languages (compare e.g. [12]). Instead we find separation results as well as encodability results with respect to very different conditions, which naturally leads to incomparable results. Among these conditions, a widely used criterion is *full abstraction*, i.e. the preservation and reflection of equivalences associated to the two compared languages. There are lots of different equivalences in the range of π-calculus variants. Since full abstraction depends, by definition, strongly on the chosen equivalences, a variation in the respective choice may change an encodability result into a separation result, or vice versa [7]. Unfortunately, there is neither a common agreement about what kinds of equivalence are well suited for language comparison—again, the results are often incomparable. To overcome these problems, and to form a more robust and uniform approach for language comparison, Gorla [6] identifies five criteria as being well suited for separation as well as encodability results. By now these criteria are widely-tested (see e.g. [5]). Here, we rely on these criteria to measure the quality of encodings between variants of the π-calculus. Compositionality and name invariance stipulate structural conditions on a valid encoding. Operational correspondence requires that a valid encoding preserves and reflects the executions of a source term. Divergence reflection states that a valid encoding shall not exhibit divergent behaviour, unless it was already present in the source term. Finally, success sensitiveness requires that a source term and its encoding have exactly the same potential to reach a successful state.

Previous Results. The question about the encodability of the match prefix is not a new one. In [17] Philips and Vigliotti proposed an encoding within the mobile ambient calculus ([3]). The π-calculus as target language was considered by Carbone and Maffeis. They proved in [2] that there exists no encoding of the π-calculus into the π-calculus (with only guarded choice and) without the match prefix. However the quality criteria used in [2] are more restrictive than the criteria here. In particular they assume that visible communication links, i.e. observables, are preserved and reflected by the encoding, i.e. a source term and its encoding must have the same observables. This criterion is very limiting (i.e. strict) even for an encoding between two variants of the same calculus. Thus, by using weaker quality criteria, we strengthen the separation result presented in [2]. Note that we use [2] as a base and starting point for our result. We discuss the differences to the proofs of [2] in Section 5.1. In the same paper Carbone and Maffeis show that the match prefix can be encoded by polyadic synchronisation. Another positive result for a variant of the π-calculus is presented by Vivas in [20]. There, a modified version of the π-calculus with a new operator, called blocking, is used to encode the match prefix. In [1] the input prefix of the π-calculus is replaced by a selective input that allows for communication only if the transmitted value is contained in a set of names specified in the selective input prefix. Accordingly selective input can be used as conditional guard, which can replace the match prefix. We discuss these encoding approaches and how they are related to our separation result in Section 5.2.

Overview. We start with an introduction of the considered variants of the π-calculus in §2. Then §3 introduces the framework of [6] to measure the quality of an encoding. Our separation result is presented in §4. In §5 we discuss the relation of our result to related work. We conclude with §6. The missing proofs and additional material can be found in [16].

2 The Pi-Calculus

Within this paper we compare two different variants of the π-calculus —the full π-calculus with (free choice and) the match prefix (π) and its variant without the match prefix ($\pi^{\not=}$)—as they are described e.g. in [9, 8].

Let $\mathcal{N}$ denote a countably infinite set of names and $\overline{\mathcal{N}}$ the set of co-names, i.e. $\overline{\mathcal{N}} = \{ \overline{n} \mid n \in \mathcal{N} \}$. We use lower case letters $a, a', a_1, \ldots, x, y, \ldots$ to range over names. Moreover let $\mathcal{N}^k$ denote the set of vectors of names of length k. Let $\mathcal{N}^*$ be the set of finite vectors of names. And let $(\tilde{x})_i = x_i$ whenever $\tilde{x} = x_1, \ldots, x_n$ and $1 \leq i \leq n$. For simplicity we adapt some set notations to deal with vectors of names, e.g. $|\tilde{x}|$ is the length of the vector $\tilde{x}$, $a \in \tilde{x}$ holds if the name a occurs in the vector $\tilde{x}$, and $\tilde{x} \cap \tilde{y} = \emptyset$ holds if the vectors $\tilde{x}$ and $\tilde{y}$ do not share a name.

Definition 1 (Syntax)**.** The set of process terms of the *full π-calculus*, denoted by $\mathcal{P}$, is given by

$$\begin{array}{c} P ::= 0 \quad | \quad x(z).P \quad | \quad \overline{x}\langle y \rangle.P \quad | \quad \tau.P \quad | \quad [a=b]P \quad | \\ P_1 + P_2 \quad | \quad P_1 \mid P_2 \quad | \quad (\nu z)P \quad | \quad !P \quad | \quad \checkmark \end{array}$$

where $a, b, x, y, z \in \mathcal{N}$. The processes of its subcalculus $\pi^{\not=}$, denoted by $\mathcal{P}^{\not=}$, are given by the same grammar without the match prefix $[a=b]P$.

The term $\checkmark$ denotes *success* (or *successful termination*). It is introduced in order to compare the abstract behaviour of terms in different process calculi as described in Section 3. The interpretation of the remaining operators is as usual. Sometimes we denote the a and b in $[a=b]P$ as *match variables*. We use $P, P', P_1, \ldots, Q, R, \ldots$ to range over processes. Let $\mathbf{fn}(P)$, $\mathbf{bn}(P)$, and $\mathbf{n}(P)$ denote the sets of *free names* in P, *bound names* in P, and all *names* occurring in P, respectively. Their definitions are completely standard, i.e. names are bound by restriction and as parameter of input and $\mathbf{n}(P) = \mathbf{fn}(P) \cup \mathbf{bn}(P)$ for all P.

We use σ, σ', σ_1, ... to range over substitutions. A substitution is a finite mapping from names to names defined by a set $\{ \{y_1/x_1\}, \ldots, \{y_n/x_n\} \}$ of renamings, where the $x_1, \ldots, x_n$ are pairwise distinct. $\{ \{y_1/x_1\}, \ldots, \{y_n/x_n\} \}(P)$ is defined as the result of simultaneously replacing all free occurrences of x_i by y_i for $i \in \{ 1, \ldots, n \}$, possibly applying alpha-conversion to avoid capture or name clashes. For all names $\mathcal{N} \setminus \{ x_1, \ldots, x_n \}$ the substitution behaves as the identity mapping, i.e. as empty substitution. We naturally extend substitutions to co-names, i.e. $\forall \sigma : \mathcal{N} \to \mathcal{N} \,.\, \forall n \in \mathcal{N} \,.\, \sigma(\overline{n}) = \overline{\sigma(n)}$.

As suggested in [6] we use a *reduction semantics* to reason about the behaviour of π and $\pi^{\not=}$. The *reduction semantics* of π and $\pi^{\not=}$ are jointly given by the transition rules

$$\tau.P \longmapsto P \qquad \overline{x}\langle y \rangle.P + P' \mid x(z).Q + Q' \longmapsto P \mid \{ y/z \} Q \qquad \frac{P \longmapsto P'}{P + Q \longmapsto P'}$$

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

where *structural congruence*, denoted by $\equiv$, is the least congruence given by the rules:

$$\begin{array}{rlcc}
P \equiv Q & \text{if } P \equiv_\alpha Q & [a=a]\,P \equiv P & !P \equiv P \mid !P \\
P + 0 \equiv P & & P + Q \equiv Q + P & P + (Q+R) \equiv (P+Q) + R \\
P \mid 0 \equiv P & & P \mid Q \equiv Q \mid P & P \mid (Q \mid R) \equiv (P \mid Q) \mid R \\
(\nu z)\,0 \equiv 0 & & (\nu z)\,(\nu w)\,P \equiv (\nu w)\,(\nu z)\,P & (\nu z)\,(P \mid Q) \equiv P \mid (\nu z)\,Q \quad \text{if } z \notin \mathbf{fn}(P)
\end{array}$$

Here $P \equiv_\alpha Q$, where $\equiv_\alpha$ denotes alpha-conversion, holds if Q can be obtained from P by renaming bound names in P, silently avoiding name clashes. Note that the structural congruence rule $[a=a]\,P \equiv P$ can be applied only in the full π-calculus. It is this structural congruence rule (in combination with the last transition rule) that defines the semantics of the match prefix. However we can similarly define the semantics of the match prefix with the reduction rule $[a=a]\,P \longmapsto P$ without any influences on our results. A reduction step $P \longmapsto P'$ then denotes either a communication between an input and output on the same link or an internal step. Let $P \longmapsto$ (and $P \not\longmapsto$) denote the existence (and non-existence) of a step from P, i.e. there is (no) P' such that $P \longmapsto P'$. Moreover, let $\Longmapsto$ be the reflexive and transitive closure of $\longmapsto$. We write $P \longmapsto^\omega$ if P can perform an infinite sequence of reduction steps. A sequence of reduction steps starting in a term P is called an *execution* of P. An execution is either finite, as $P_0 \longmapsto P_1 \longmapsto \ldots \longmapsto P_n$, or infinite. A finite execution $P_0 \Longmapsto P_n$ is *maximal* if it cannot be further extended, i.e. if $P_n \not\longmapsto$, otherwise it is *partial*.

Traditionally a process term is considered as successful if it has an unguarded occurrence of success (see e.g. [6]). This is usually formalised as $\exists P' \,.\, P \equiv \checkmark \mid P'$. Because of free choice, we have to adapt the usual definitions of the reachability of success to deal with arbitrary nestings of choice and parallel composition. To do so we recursively define the notion of unguarded subterms.

Definition 2 (Unguarded Subterms)**.** Let $P \in \mathscr{P}$ or $P \in \mathscr{P}^{\asymp}$. The *set of unguarded subterms of* P, denoted by $\mathbf{ungSub}(P)$, is recursively defined as:

$$\begin{cases}
\{\, P \,\} \cup \mathbf{ungSub}(Q) & \text{, if } P = [a=a]\,Q \\
\{\, P \,\} \cup \mathbf{ungSub}(Q_1) \cup \mathbf{ungSub}(Q_2) & \text{, if } P = Q_1 + Q_2 \vee P = Q_1 \mid Q_2 \\
\{\, P \,\} \cup \mathbf{ungSub}(Q) & \text{, if } P = (\nu z)\,Q \vee P = !Q \\
\{\, P \,\} & \text{, otherwise}
\end{cases}$$

Note that the sets of unguarded subterms can differ for structural congruent terms. Consider for example $\mathbf{ungSub}((\nu z)\,\overline{z}\langle z\rangle.0) = \{\, (\nu z)\,\overline{z}\langle z\rangle.0, \overline{z}\langle z\rangle.0 \,\}$ but $\mathbf{ungSub}\big((\nu z')\,\overline{z'}\langle z'\rangle.0\big) = \big\{\, (\nu z')\,\overline{z'}\langle z'\rangle.0, \overline{z'}\langle z'\rangle.0 \,\big\}$ or $\mathbf{ungSub}(\checkmark) = \{\, \checkmark \,\}$ but $\mathbf{ungSub}(\checkmark + 0) = \{\, \checkmark + 0, \checkmark, 0 \,\}$. Similarly, injective substitutions do not distribute over unguarded subterms. For example $\{\, {}^{y}/_{x} \,\}(\mathbf{ungSub}((\nu x)\,\overline{x}\langle x\rangle.0)) = \{\, (\nu x)\,\overline{x}\langle x\rangle.0, \overline{y}\langle y\rangle.0 \,\}$ but $\{\, {}^{y}/_{x} \,\}(\mathbf{ungSub}((\nu z)\,\overline{z}\langle z\rangle.0)) = \{\, (\nu z)\,\overline{z}\langle z\rangle.0, \overline{z}\langle z\rangle.0 \,\}$. Moreover note that if P' is an unguarded subterm of P then also all unguarded subterms of P' are unguarded subterms of P.

Then a term is successful if it has an unguarded occurrence of success.

Definition 3 (Reachability of Success)**.** Let $P \in \mathscr{P}$ or $P \in \mathscr{P}^{\asymp}$. Then P is *successful*, denoted by $P\downarrow_\checkmark$, if $\checkmark \in \mathbf{ungSub}(P)$. P *reaches success*, denoted by $P\Downarrow_\checkmark$, if there is some Q such that $P \Longmapsto Q$ and $Q\downarrow_\checkmark$. Moreover, we write $P \Downarrow_{\checkmark!}$, if P reaches success in every finite maximal execution. Let $P\not\downarrow_\checkmark$ abbreviate $\neg\,(P\downarrow_\checkmark)$, $P\not\Downarrow_\checkmark$ abbreviate $\neg\,(P\Downarrow_\checkmark)$, and $P \not\Downarrow_{\checkmark!}$ abbreviate $\neg\,(P \Downarrow_{\checkmark!})$.

Of course, all proofs in this paper hold similarly for variants of π and $\pi^{\asymp}$ with only guarded choice and the traditional definition of a successful term.

The first quality criterion to compare process calculi presented in Section 3 is compositionality. It induces the definition of a $\pi^{\asymp}$-context parametrised on a set of names for each operator of π. A

$\pi^{\asymp}$-context $\mathsf{C}([\cdot]_1,\ldots,[\cdot]_n) : (\mathscr{P}^{\asymp})^n \to \mathscr{P}^{\asymp}$ is simply a $\pi^{\asymp}$-term with n holes. Putting some $\pi^{\asymp}$-terms $P_1,\ldots,P_n$ in this order into the holes $[\cdot]_1,\ldots,[\cdot]_n$ of the context, respectively, gives a term denoted by $\mathsf{C}(P_1,\ldots,P_n)$. Note that a context may bind some free names of $P_1,\ldots,P_n$. The arity of a context is the number of its holes. We extend the definition of unguarded subterms by the equation $\mathbf{ungSub}([\cdot]) = \{ [\cdot] \}$ to deal with contexts.

The standard notion of equivalence to compare terms of the π-calculus is bisimulation. An introduction to bisimulations in the π-calculus can be found e.g. in [9] or [19]. For our separation result we require such a standard version of reduction bisimulation, denoted by $\asymp$, on the target language, i.e. on $\pi^{\asymp}$-terms.

3 Quality of Encodings

Within this paper we analyse the existence of an encoding from π into $\pi^{\asymp}$. To measure the quality of such an encoding, Gorla [6] suggested five criteria well suited for language comparison. Accordingly, we consider an encoding to be "valid", if it satisfies Gorla's five criteria.

We call the tuple $\mathscr{L} = (\mathscr{P}, \longmapsto)$, where $\mathscr{P}$ is a set of language terms and $\longmapsto$ is a reduction semantics, a *language*. An *encoding* from $\mathscr{L}_1 = (\mathscr{P}_1, \longmapsto_1)$ into $\mathscr{L}_2 = (\mathscr{P}_2, \longmapsto_2)$ is then a tuple $(\llbracket \cdot \rrbracket, \varphi_{\llbracket\rrbracket}, \asymp)$ such that

- $\llbracket \cdot \rrbracket : \mathscr{P}_1 \to \mathscr{P}_2$ is the translating function,
- $\varphi_{\llbracket\rrbracket} : \mathscr{N} \to \mathscr{N}^k$ is a renaming policy, where $\varphi_{\llbracket\rrbracket}(u) \cap \varphi_{\llbracket\rrbracket}(v) = \emptyset$ for all $u \neq v$,
- and $\asymp$ is a behaviour equivalence on $\mathscr{L}_2$.

We call $\mathscr{L}_1$ the *source language (calculus)* and $\mathscr{L}_2$ the *target language (calculus)*. Accordingly we call the elements of $\mathscr{P}_1$ *source terms* and the elements of $\mathscr{P}_2$ *target terms*. We use $S, S', S_1, \ldots$ $(T, T', T_1, \ldots)$ to range over source (target) terms.

The main ingredient of an encoding is of course the encoding function $\llbracket \cdot \rrbracket$ that is a mapping from processes to processes. However, sometimes it is useful to be able to reserve some names to play a special role in an encoding. Since most process calculi have infinitely many names in their alphabet, it suffices to shift the set of names $\{ x_0, x_1, \ldots \}$ of the target language to the set $\{ x_n, x_{n+1}, \ldots \}$ to reserve n names. In order to incorporate such "shifts" and similar techniques, Gorla introduces a renaming policy $\varphi_{\llbracket\rrbracket}$, i.e. mapping from names to names that specifies the translation of each name of the source language into a name or vector of names of the target language. Additionally we assume the existence of a behavioural equivalence $\asymp$ on the target language that is a reduction bisimulation. Its purpose is to describe the abstract behaviour of a target process, where abstract basically means with respect to the behaviour of the source term. Therefore it should abstract from "junk" left over by the encoding.

[6] requires $\varphi_{\llbracket\rrbracket}$ to map all names to a vector of the same length since this way names are treated uniformly, i.e. source names cannot be handled differently by an encoding just because the length of the vector, to that $\varphi_{\llbracket\rrbracket}$ maps to, is different. The condition that $\varphi_{\llbracket\rrbracket}(u) \cap \varphi_{\llbracket\rrbracket}(v) = \emptyset$ for all $u \neq v$ ensures that the renaming policy does not relate unrelated source term names.

Definition 4 (Valid Encoding)**.** An encoding from $\mathscr{L}_1 = (\mathscr{P}_1, \longmapsto_1)$ into $\mathscr{L}_2 = (\mathscr{P}_2, \longmapsto_2)$ is *valid* if it satisfies:

Compositionality: For each k-ary operator $\mathtt{op}$ of $\mathscr{L}_1$ and all sets of names $N \subseteq \mathscr{N}$ there is a k-ary context $\mathsf{C}^N_{\mathtt{op}}([\cdot]_1,\ldots,[\cdot]_k)$ such that for all $S_1,\ldots,S_k \in \mathscr{P}_1$ with $\mathbf{fn}(S_1,\ldots,S_k) = N$ it holds that $\llbracket \mathtt{op}(S_1,\ldots,S_k) \rrbracket = \mathsf{C}^N_{\mathtt{op}}(\llbracket S_1 \rrbracket,\ldots,\llbracket S_k \rrbracket)$.

Name Invariance: For each S and σ it holds that

$$[\![\, \sigma(S) \,]\!] \begin{cases} = \sigma' \left([\![\, S \,]\!] \right) & \text{if } \sigma \text{ is injective} \\ \asymp \sigma' \left([\![\, S \,]\!] \right) & \text{otherwise} \end{cases}$$

where σ' is such that $\varphi_{[\![\,]\!]}(\sigma(a)) = \sigma'(\varphi_{[\![\,]\!]}(a))$ for every $a \in \mathcal{N}$.

Operational Correspondence:

Complete: For all $S \Longmapsto S'$, it holds that $[\![\, S \,]\!] \Longmapsto \asymp [\![\, S' \,]\!]$.

Sound: For all $[\![\, S \,]\!] \Longmapsto T$, there is S' such that $S \Longmapsto S'$ and $T \Longmapsto \asymp [\![\, S' \,]\!]$.

Divergence Reflection: For every S with $[\![\, S \,]\!] \longmapsto^{\omega}$, it holds that $S \longmapsto^{\omega}$.

Success Sensitiveness: For every S, it holds $S \Downarrow_{\checkmark}$ iff $[\![\, S \,]\!] \Downarrow_{\checkmark}$.

Intuitively, an encoding is compositional if the translation of an operator is similar for all its parameters. To mediate between the translations of the parameters the encoding defines a unique context for each operator, whose arity is the arity of the operator. Moreover, the context can be parametrised on the free names of the corresponding source term. Note that our result is independent of this parametrisation. In name invariance the σ' can be considered as the translation of σ. The condition $\varphi_{[\![\,]\!]}(\sigma(a)) = \sigma'(\varphi_{[\![\,]\!]}(a))$ ensures that $\varphi_{[\![\,]\!]}$ introduces no additional renamings between (parts of) translations of source term names. Of course σ' cannot affect reserved names, i.e. for all names x in the domain of σ' there is a source term name a such that $x \in \varphi_{[\![\,]\!]}(a)$. Operational correspondence consists of a soundness and a completeness condition. *Completeness* requires that every execution of a source term can be simulated by its translation, i.e. the translation does not omit any execution of the source term. *Soundness* requires that every execution of a target term corresponds to some execution of the corresponding source term, i.e. the translation does not introduce new executions. Note that the definition of operational correspondence relies on the equivalence $\asymp$ to get rid of junk possibly left over within executions of target terms. An encoding reflects divergence if it does not introduce divergent executions. The last criterion states that an encoding preserves the behaviour of the source term if it and its corresponding target term answer the tests for success in exactly the same way.

Success sensitiveness only links the behaviours of source terms and their literal translations but not of their continuations. To do so, Gorla relates success sensitiveness and operational correspondence by requiring that $\asymp$ never relates two processes that differ in the possibility to reach success. More precisely $\asymp$ *respects success* if, for every P and Q with $P \Downarrow_{\checkmark}$ and $Q \not\Downarrow_{\checkmark}$, it holds that $P \not\asymp Q$. By [6] a "good" equivalence $\asymp$ is often defined in the form of a barbed equivalence (as described e.g. in [10]) or can be derived directly from the reduction semantics and is often a congruence, at least with respect to parallel composition. For the separation results presented in this paper, we require only that $\asymp$ is a success respecting reduction bisimulation, i.e. for every $T_1, T_2 \in \mathcal{P}_2$ such that $T_1 \asymp T_2$, $T_1 \Downarrow_{\checkmark}$ iff $T_2 \Downarrow_{\checkmark}$ and for all $T_1 \Longmapsto_2 T_1'$ there exists a T_2' such that $T_2 \Longmapsto_2 T_2'$ and $T_1' \asymp T_2'$.

4 The Match Prefix and the Pi-Calculus

Our separation result strongly rests on the criteria compositionality and success sensitiveness. We also make use of name invariance. But, as we claim, name invariance is not crucial for the proof. Name invariance defines how a valid encoding has to deal with substitutions. This is used to simplify the argumentation in our proof as explained below. The last criterion states that source and target terms are related by their ability to reach success. If we compare π and $\pi^{\not=}$ we observe a difference with respect to successful terms and substitutions. In π a substitution can change the state of a process from unsuccessful to successful. Consider for example the term $[a = b] \checkmark$ and a substitution σ such that $\sigma(a) = \sigma(b)$.

The only occurrence of success in $[a = b]\checkmark$ is guarded by a match prefix and thus $([a = b]\checkmark)\not\Downarrow_{\checkmark}$. But $\sigma([a = b]\checkmark) = [\sigma(a) = \sigma(b)]\checkmark$ and thus $\sigma([a = b]\checkmark)\downarrow_{\checkmark}$. In $\pi^{\asymp}$, because there is no match prefix, a substitution cannot turn an unsuccessful state into a successful state.

Lemma 5. *Let $T \in \mathscr{P}^{\asymp}$. Then $T\downarrow_{\checkmark}$ **iff** $\forall \sigma : \mathscr{N} \to \mathscr{N} \, . \, \sigma(T)\downarrow_{\checkmark}$.*

In both calculi substitutions may allow us to reach success by enabling a communication step. To do so it has to unify two free names that are the links of an unguarded input and an unguarded output. In the case of $\pi^{\asymp}$ the enabling of such a new communication step is indeed the only possibility for a substitution to influence the reachability of success. More precisely, if in $\pi^{\asymp}$ a substitution σ allows to reach success, i.e. if $\sigma(T)\Downarrow_{\checkmark}$ but $T\not\Downarrow_{\checkmark}$, then there is a derivative of T in which σ unifies the free link names of an input and an output guard and thus enables a new communication step.

Lemma 6. *Let $T \in \mathscr{P}^{\asymp}$ and $\sigma : \mathscr{N} \to \mathscr{N}$ such that $\sigma(T)\Downarrow_{\checkmark}$ but $T\not\Downarrow_{\checkmark}$. Then:*

$$\begin{aligned}&\exists T', T_1, T_2, T_3, T_4 \in \mathscr{P}^{\asymp} \, . \, \exists y \in \mathscr{N} \, . \, \exists a, b \in \mathbf{fn}(T) \, . \\ &\quad T \Longmapsto\equiv T' \, \wedge \, a, b \notin \mathbf{bn}(T') \, \wedge \, (T_1 \mid T_2) \in \mathbf{ungSub}(T') \, \wedge \, a(y).T_3 \in \mathbf{ungSub}(T_1) \\ &\quad \wedge \, \overline{b}\langle y\rangle.T_4 \in \mathbf{ungSub}(T_2) \, \wedge \, \sigma(a) = \sigma(b) \, \wedge \, a \neq b\end{aligned}$$

In the following proofs we often use the term $[a = b]\checkmark$ or a variant of this term as counterexample. To reason about the encoding of this term we analyse the context $\mathsf{C}_{[a=b]}^{N \cup \{a,b\}}([\cdot])$ that is introduced according to compositionality to translate $[a = b]$. Note that this context is parameterised on $N \cup \{\, a, b \,\}$, which is the set of free names of the encoded term. For example in the case of $[a = b]\checkmark$ the set of free names contains only a and b, i.e. $N = \emptyset$. Moreover a, b and the continuation of the match prefix are parameters of this context. First we show that this context cannot reach success on its own, i.e. without a term in its hole.

Lemma 7. *Let $(\llbracket \cdot \rrbracket, \varphi_{\llbracket\rrbracket}, \asymp)$ be a valid encoding from π into $\pi^{\asymp}$. Let $N \subseteq \mathscr{N}$ be a finite set of names, $a, b \in \mathscr{N}$ be names such that $a \neq b$, and $\mathsf{C}_{[a=b]}^{N \cup \{a,b\}}([\cdot])$ be the context that is introduced by $\llbracket \cdot \rrbracket$ to encode the match prefix $[a = b]$. Then $\mathsf{C}_{[a=b]}^{N \cup \{a,b\}}([\cdot])$ cannot reach success on its own, i.e. $\mathsf{C}_{[a=b]}^{N \cup \{a,b\}}([\cdot])\not\Downarrow_{\checkmark}$.*

Moreover the context introduced to encode the match prefix has to ensure that its hole, i.e. the respective encoding of the continuation of the match prefix, is initially guarded and cannot be unguarded by the context on its own.

Lemma 8. *Let $(\llbracket \cdot \rrbracket, \varphi_{\llbracket\rrbracket}, \asymp)$ be a valid encoding from π into $\pi^{\asymp}$. Let $N \subseteq \mathscr{N}$ be an arbitrary finite set of names, $a, b \in \mathscr{N}$ be arbitrary names such that $a \neq b$, and $\mathsf{C}_{[a=b]}^{N \cup \{a,b\}}([\cdot])$ be the context that is introduced by $\llbracket \cdot \rrbracket$ to encode the match prefix $[a = b]$. Then $\mathsf{C}_{[a=b]}^{N \cup \{a,b\}}([\cdot])$ cannot unguard its hole, i.e. $\mathsf{C}_{[a=b]}^{N \cup \{a,b\}}([\cdot]) \Longmapsto \mathsf{C}'([\cdot])$ implies $[\cdot] \notin \mathbf{ungSub}(\mathsf{C}'([\cdot]))$ for all $\mathsf{C}'([\cdot]) : \mathscr{P}^{\asymp} \to \mathscr{P}^{\asymp}$.*

Next we combine our knowledge of the context $\mathsf{C}_{[a=b]}^{N \cup \{a,b\}}([\cdot])$ and the relationship between substitutions and the reachability of success in $\pi^{\asymp}$ as stated in Lemma 6. We show that a match prefix $[a = b]$ has to be translated into two communication partners, i.e. an input and an output, on the translations of a and b. Intuitively such a communication is the only way to simulate the test for equality of names that is performed by $[a = b]$. Moreover we derive that the respective links of the communication partner have to be free in the context $\mathsf{C}_{[a=b]}^{N \cup \{a,b\}}([\cdot])$. Intuitively they have to be free, because otherwise no substitution can unify them. Consider for example the term $\overline{x}\langle a\rangle.0 \mid x(b).[a = b]\checkmark$. In order to reach success the term first communicates the name a on x. This communication leads to a substitution of the name b by the received value a in the continuation of the input guarded subterm. Only this substitution allows to unguard the only occurrence of success. Hence, to simulate such a behaviour of source terms, the encoding has to translate match prefixes into communication partners—to simulate the test for equality—and the links of these communication partners have to be free—to allow for substitutions induced by communication steps. Note that name invariance allows us to ignore a surrounding communication—as the step on link x in the example $\overline{x}\langle a\rangle.0 \mid x(b).[a=b]\checkmark$—and to concentrate directly on the induced substitution.

To avoid the use of the criterion name invariance it suffices to show that the encodings of $\overline{x}\langle a\rangle.0 \mid x(b).[a=b]\checkmark$ and $\overline{x}\langle b\rangle.0 \mid x(b).[a=b]\checkmark$ differ only by a substitution of (parts of) the translations of a and b[2] and that the contexts introduced to encode outputs and inputs cannot lead to success themselves, i.e. that the encoding of $\overline{x}\langle a\rangle.0 \mid x(b).[a=b]\checkmark$ reaches success iff the encoding of $[a=b]\checkmark$ is unguarded. This suffices to reconstruct the substitution and the conditions on this substitution that are used in Lemma 9 and to prove Lemma 6 and Lemma 12 w.r.t. such substitutions. The remaining proofs remain the same.

Note that the appendix provides a more formal formulation of the next lemma.

Lemma 9. *Let $(\llbracket\cdot\rrbracket, \varphi_{\llbracket\rrbracket}, \asymp)$ be a valid encoding from π into $\pi^{\not\asymp}$. Let $N \subseteq \mathcal{N}$ be an arbitrary finite set of names, $a,b \in \mathcal{N}$ be arbitrary names such that $a \neq b$, and $\mathsf{C}_{[a=b]}^{N\cup\{a,b\}}([\cdot])$ be the context that is introduced by $\llbracket\cdot\rrbracket$ to encode the match prefix $[a=b]$. Then there is some $i \in \left\{\, 1,\ldots,\left|\varphi_{\llbracket\rrbracket}(a)\right| \,\right\}$ such that $\mathsf{C}_{[a=b]}^{N\cup\{a,b\}}([\cdot])$ reaches a state with an unguarded and free output and input on the links $\left(\varphi_{\llbracket\rrbracket}(a)\right)_i$ and $\left(\varphi_{\llbracket\rrbracket}(b)\right)_i$. Moreover $\mathsf{C}_{[a=b]}^{N\cup\{a,b\}}([\cdot])$ cannot unguard its hole until a substitution unifies these two links.*

A very important consequence of the lemma above is the existence of the index i for all contexts $\mathsf{C}_{[a=b]}^{N\cup\{a,b\}}([\cdot])$ regardless of N and of the terms that may be inserted in the hole. Note that the "there is some" does not necessarily imply that there is just one such i. If the renaming policy splits up a source term name into several target term names then different parts of this vector can be used to simulate the test for equality. However, the above lemma states that there is at least one such i, i.e. at least one part of the translation of source term names is used to implement the required communication partners.

To derive the separation result we need a counterexample that combines two match prefixes in parallel. Therefore we need some information on the context $\mathsf{C}_{\mid}^{N}([\cdot]_1\,;[\cdot]_2)$ that is introduced by $\llbracket\cdot\rrbracket$ according to compositionality to translate the parallel operator. Note that this context is parameterised on the set N that consists of the free names of the two parallel components that should be encoded. Moreover the two holes serve as placeholders for the encoding of the left and the right hand side of the source term. Similar to the context introduced to encode the match prefix, the context that is introduced to encode the parallel operator cannot reach success on its own, i.e. $\mathsf{C}_{\mid}^{N}([\cdot]_1\,;[\cdot]_2)\not\Downarrow_{\checkmark}$.

Lemma 10. *Let $(\llbracket\cdot\rrbracket, \varphi_{\llbracket\rrbracket}, \asymp)$ be a valid encoding from π into $\pi^{\not\asymp}$. Let $N \subseteq \mathcal{N}$ be a finite set of names and $\mathsf{C}_{\mid}^{N}([\cdot]_1\,;[\cdot]_2)$ be the context introduced by $\llbracket\cdot\rrbracket$ to encode the parallel operator. Then, $\mathsf{C}_{\mid}^{N}([\cdot]_1\,;[\cdot]_2)\not\Downarrow_{\checkmark}$.*

But in contrast to the context introduced in order to encode the match prefix the context $\mathsf{C}_{\mid}^{N}([\cdot]_1\,;[\cdot]_2)$ has always, i.e. regardless of a substitution, to unguard its holes on its own.

Lemma 11. *Let $(\llbracket\cdot\rrbracket, \varphi_{\llbracket\rrbracket}, \asymp)$ be a valid encoding from π into $\pi^{\not\asymp}$. Let $N \subseteq \mathcal{N}$ be a finite set of names and $\mathsf{C}_{\mid}^{N}([\cdot]_1\,;[\cdot]_2)$ be the context introduced by $\llbracket\cdot\rrbracket$ to encode the parallel operator. Then there is some $T \in \mathcal{P}^{\not\asymp}$ such that $\mathsf{C}_{\mid}^{N}([\cdot]_1\,;[\cdot]_2) \Longmapsto T$ and $[\cdot]_1, [\cdot]_2 \in$ **ungSub**(T).*

[2]Unfortunately, because of the formulation of compositionality that allows for the contexts to depend on the free names of the term, this task is technically elaborate.

Then we need to show that the context $\mathsf{C}^{N}_{|}([\cdot]_1 ; [\cdot]_2)$ cannot bind the names that are used by the context $\mathsf{C}^{N \cup \{a,b\}}_{[a=b]}([\cdot])$ to simulate the test for equality. As in the above example $\overline{x}\langle a \rangle.0 \mid x(b).[a=b]\checkmark$, a communication step can unify at runtime the variables of a match prefix. Such a communication step naturally transmits the value for the match variables over a parallel operator, because communication is always between two communication partners that are composed in parallel. If this value is restricted on either side of the parallel operator the communication could not lead to the required unification. The match variables would still be considered as different and the match prefix as not satisfied. Thus for example neither $(\nu a)\,(\overline{x}\langle a \rangle.0) \mid x(b).[a=b]\checkmark$ nor $\overline{x}\langle a \rangle.0 \mid (\nu a)\,(x(b).[a=b]\checkmark)$ reach success although in both cases the communication on x is still possible. Of course the term $(\nu a)\,(\overline{x}\langle a \rangle.0 \mid x(b).[a=b]\checkmark)$ reaches success. But for cases like this we can construct larger counterexamples as $(\nu a)\,(\overline{x}\langle a \rangle.0 \mid 0) \mid x(b).[a=b]\checkmark$ and to analyse the source term, in order to examine the places at which such a restriction would be allowed, violates the idea of a compositional encoding. Again name invariance allows us to ignore the communication on x and to directly concentrate on the induced substitution.

Lemma 12. *Let $(\llbracket \cdot \rrbracket, \varphi_{\llbracket\rrbracket}, \asymp)$ be a valid encoding from π into $\pi^{\not\asymp}$. Let $N \subseteq \mathcal{N}$ be a finite set of names and $\mathsf{C}^{N}_{|}([\cdot]_1 ; [\cdot]_2)$ be the context that is introduced by $\llbracket \cdot \rrbracket$ to encode the parallel operator. Then $\left(\varphi_{\llbracket\rrbracket}(a)\right)_i \in \mathbf{fn}\left(\mathsf{C}^{N}_{|}(\llbracket P \rrbracket ; \llbracket Q \rrbracket)\right)$ for all $P, Q \in \mathscr{P}$ and all $a \in \mathbf{fn}(P \mid Q)$, where $i \in \{\, 1, \ldots, \left|\varphi_{\llbracket\rrbracket}(a)\right| \,\}$ is the index that exists according to Lemma 9.*

Finally we show that there is no valid encoding from π into $\pi^{\not\asymp}$, by assuming the contrary and deriving a contradiction. As already mentioned, we use a counterexample that consists of two parallel composed match prefixes. More precisely we use $[a=b]\checkmark \mid [b=a]\checkmark$, i.e. swap the match variables on the right side. Intuitively the contradiction is derived as follows: Since the context $\mathsf{C}^{N \cup \{a,b\}}_{[a=b]}([\cdot])$ translates the match variables into free links of unguarded communication partners and because of the swapping of the matching variables on the right side, the parallel composition of the two variants of the context $\mathsf{C}^{N \cup \{a,b\}}_{[a=b]}([\cdot])$—that are necessary to encode the counterexample—enable wrong communication steps between a communication partner from the left $\mathsf{C}^{N \cup \{a,b\}}_{[a=b]}([\cdot])$ and a communication partner from the right $\mathsf{C}^{N \cup \{a,b\}}_{[b=a]}([\cdot])$. We denote such a communication step as wrong, because in this case the communication cannot lead to the unguarding of the encoded continuation $\llbracket \checkmark \rrbracket$ without violating success sensitiveness. In order to reach success, the source term needs a substitution σ that unifies the match variables. Unfortunately, the same wrong communication can consume one of the communication partners in $\llbracket \sigma\,([a=b]\checkmark \mid [b=a]\checkmark) \rrbracket$ that is necessary to unguard the encoded continuation. A restoration of this communication partner leads by symmetry to divergence which violates the divergence reflection criterion. But without the possibility of a restoration, the wrong communication leads to an unsuccessful execution of $\llbracket \sigma\,([a=b]\checkmark \mid [b=a]\checkmark) \rrbracket$. This execution violates the combination of success sensitiveness and operational soundness.

Theorem 1. *There is no valid encoding from π into $\pi^{\not\asymp}$.*

Proof Sketch. Assume the contrary, i.e. assume that there is a valid encoding $(\llbracket \cdot \rrbracket, \varphi_{\llbracket\rrbracket}, \asymp)$ from π into $\pi^{\not\asymp}$. Consider the term $S = [a=b]\checkmark \mid [b=a]\checkmark$ and a substitution $\sigma : \mathcal{N} \to \mathcal{N}$ such that $\sigma(a) = \sigma(b)$.

By Lemma 11, there is some T such that $\llbracket S \rrbracket \Longmapsto T$ and $\mathsf{C}^{\{a,b\}}_{[a=b]}(\llbracket \checkmark \rrbracket), \mathsf{C}^{\{a,b\}}_{[b=a]}(\llbracket \checkmark \rrbracket) \in \mathbf{ungSub}(T)$. Because $\mathbf{fn}([a=b]P) = \mathbf{fn}([b=a]P)$ for all P, $\mathsf{C}^{\{a,b\}}_{[a=b]}([\cdot]) = \{\, \varphi_{\llbracket\rrbracket}(b)/\varphi_{\llbracket\rrbracket}(a), \varphi_{\llbracket\rrbracket}(a)/\varphi_{\llbracket\rrbracket}(b) \,\}\left(\mathsf{C}^{\{a,b\}}_{[b=a]}([\cdot])\right)$. By Lemma 9, then there are $i \in \{\, 1, \ldots, \left|\varphi_{\llbracket\rrbracket}(a)\right| \,\}$, T', $\mathsf{C}_1([\cdot])$, $\mathsf{C}_2([\cdot])$, $\mathsf{C}_3([\cdot])$, and $\mathsf{C}_4([\cdot])$ such that $\llbracket S \rrbracket \equiv\!\Longmapsto T'$ and T' has an unguarded input as well as an unguarded output on both of the channels $\left(\varphi_{\llbracket\rrbracket}(a)\right)_i$ and $\left(\varphi_{\llbracket\rrbracket}(b)\right)_i$.

By name invariance, $\llbracket\, \sigma(S) \,\rrbracket \asymp \sigma'(\llbracket\, S \,\rrbracket) = \sigma'\left(\mathsf{C}_{\mid}^{\{a,b\}}\left(\mathsf{C}_{[a=b]}^{\{a,b\}}(\llbracket \checkmark \rrbracket) ; \mathsf{C}_{[b=a]}^{\{a,b\}}(\llbracket \checkmark \rrbracket)\right)\right)$, where $\varphi_{\llbracket\rrbracket}(\sigma(n)) = \sigma'(\varphi_{\llbracket\rrbracket}(n))$ for every $n \in \mathcal{N}$. Hence $\sigma'(\llbracket\, S \,\rrbracket) \equiv\Longmapsto \sigma'(T')$, i.e. the inputs on the channels $\left(\varphi_{\llbracket\rrbracket}(a)\right)_i$ and $\left(\varphi_{\llbracket\rrbracket}(b)\right)_i$ can communicate between the two instances of the context $\mathsf{C}_{[\cdot=\cdot]}^{\{a,b\}}(\cdot)$ in $\sigma'(T')$. By the argumentation above these communications, i.e. there an input from the left $\mathsf{C}_{[a=b]}^{\{a,b\}}(\llbracket \checkmark \rrbracket)$ interacts with an output from the right $\mathsf{C}_{[b=a]}^{\{a,b\}}(\llbracket \checkmark \rrbracket)$ or vice versa, cannot lead to the unguarding of $\llbracket \checkmark \rrbracket$. Note that if either the left $\mathsf{C}_{[a=b]}^{\{a,b\}}(\llbracket \checkmark \rrbracket)$ or the right $\mathsf{C}_{[b=a]}^{\{a,b\}}(\llbracket \checkmark \rrbracket)$ restores a wrongly consumed input term or output term on $\left(\varphi_{\llbracket\rrbracket}(a)\right)_i$ or $\left(\varphi_{\llbracket\rrbracket}(b)\right)_i$ then, because $\mathsf{C}_{[a=b]}^{\{a,b\}}([\cdot]) = \{\, \varphi_{\llbracket\rrbracket}(b)/\varphi_{\llbracket\rrbracket}(a), \varphi_{\llbracket\rrbracket}(a)/\varphi_{\llbracket\rrbracket}(b) \,\}\left(\mathsf{C}_{[b=a]}^{\{a,b\}}([\cdot])\right)$, there is an execution there the other side also restores the corresponding counterpart. This leads back to the state before the respective communication step between the left $\mathsf{C}_{[a=b]}^{\{a,b\}}(\llbracket \checkmark \rrbracket)$ and the right $\mathsf{C}_{[b=a]}^{\{a,b\}}(\llbracket \checkmark \rrbracket)$ and thus to a divergent execution. The same holds if the context $\mathsf{C}_{\mid}^{\{a,b\}}([\cdot]_1 ; [\cdot]_2)$ restores such an input term or output term. But since $\sigma(S)$ has no divergent execution and because of divergence reflection, a divergent execution of $\llbracket\, \sigma(S) \,\rrbracket$ violates our assumption that $(\llbracket \cdot \rrbracket, \varphi_{\llbracket\rrbracket}, \asymp)$ is a valid encoding. Thus $\sigma'(\llbracket\, S \,\rrbracket)$ cannot restore a wrongly consumed input term or output term on $\left(\varphi_{\llbracket\rrbracket}(a)\right)_i$ or $\left(\varphi_{\llbracket\rrbracket}(b)\right)_i$.

By Lemma 9, only a communication between the terms on channels $\left(\varphi_{\llbracket\rrbracket}(a)\right)_i$ and $\left(\varphi_{\llbracket\rrbracket}(b)\right)_i$ can unguard the continuation $\llbracket \checkmark \rrbracket$. Hence there is a finite maximal execution of $\llbracket\, \sigma(S) \,\rrbracket$ in which the continuation $\llbracket \checkmark \rrbracket$ is never unguarded. Thus, by Lemma 7 and Lemma 10, no success is reached in this execution, i.e. $\llbracket\, \sigma(S) \,\rrbracket \not\Downarrow_{\checkmark!}$. But $\sigma(S) \Downarrow_{\checkmark!}$ implies $\llbracket\, \sigma(S) \,\rrbracket \Downarrow_{\checkmark!}$. This is a contradiction. ☐

5 Discussion

As mentioned above, also Carbone and Maffeis show in [2] that the match prefix cannot be encoded within the π-calculus. Moreover there are different encodings of the match prefix in modified variants and extensions of the π-calculus. In this section we discuss the relation between these results and our separation result.

5.1 The Match Prefix is a Native Operator of the Pi-Calculus

If we compare the approach in [2] with ours, we observe that the considered variants of the π-calculus are different. We consider the full π-calculus and its variant without the match prefix as source and target language. In the literature there are different variants called “full” π-calculus. We decide on the most general of these variants. In particular we consider a variant of the π-calculus with free choice whereas [2] allow only guarded choice in their target language. Note that the source language considered in [2] is an asynchronous variant of the π-calculus, i.e. is less expressive than the source language considered here [11, 14, 15]. However, the only (counter)examples we use here are of the form $[a = b]\,X$, or $[a = b]\,X \mid [b = a]\,X$ where X is a combination of $\checkmark$, 0, P, and parallel composition for an arbitrary P with a fixed set of free names. Thus, our separation result remains valid if we change the source language to the asynchronous variant of the π-calculus without choice that is used in [2]. Our target language is also more expressive, because we do not restrict it to guarded choice. More precisely, in [2] the π-calculus with guarded mixed choice is used. Accordingly, the current result can be considered stronger. However, concentrating only on guarded choice is commonly accepted and, more importantly, it might be easy to adapt the proof in [2] to the more expressive target language.

Contribution 1. The main difference between the two approaches are the quality criteria, i.e. in the conditions that are assumed to hold for all valid encodings. Similar to [11], Carbone and Maffeis require that an encoding must be uniform and reasonable. By [2] an encoding $[\![\cdot]\!]$ is *uniform* if it translates the parallel operator homomorphically, i.e. $[\![P \mid Q]\!] = [\![P]\!] \mid [\![Q]\!]$, and if it respects permutations on free names, i.e. for all σ there is some θ such that $[\![\sigma(P)]\!] = \theta([\![P]\!])$. A *reasonable* semantics, by [2], is one which distinguishes two processes P and Q whenever there exists a maximal execution of Q in which the observables are different from the observables in any maximal execution of P. Furthermore they require that an encoding should be able to distinguish deadlocks from livelocks, which is comparable to divergence reflection.

In contrast to uniformity, name invariance relates the substitution on the source term names with its translation on target term names. Already [6] points out that name invariance is a more complex requirement than the above condition; but [6] also argues that it is rather more detailed than more demanding. Moreover we claim that name invariance is not crucial for the above separation result. The first condition of uniformity is a strictly stronger requirement than compositionality for the parallel operator as it is discussed for instance in [13]. However the proof in [2] does not use the homomorphic translation of the parallel operator.

The criterion on the reasonable semantics used in [2] is even more demanding than the first part of uniformity. It states that a source term and its encoding reach exactly the same observables. It completely ignores the possibility to translate a source term name into a sequence of names or to simulate a source term observable by a set of target term observables even if there is a bijective mapping between an observable and its translation. The proof in [2] makes strongly use of this criterion; exploiting the fact that the match variables are free in the match prefix. Gorla suggests success sensitiveness and operational correspondence instead. Note that we use operational correspondence—or more precisely soundness—only in the last step of the proof to argument that if a source term reaches success in all finite maximal executions its encoding does alike. Hence, for the presented case, the combination of operational soundness and success sensitiveness is a considerably weaker requirement than the variant of reasonableness.

Overall we conclude that, because of the large difference between success sensitiveness and the variant of reasonableness considered in [2], our set of criteria is considerably weaker and thus the presented result is strictly stronger.

Contribution 2. The proof in [2] is, due to the stricter criteria, shorter and easier to follow than ours. But it also reveals less information on the reason for the separation result. In contrast, the presented approach reflects the intuition that communication is close to the behaviour of the match prefix. We show that among the native operators of the π-calculus input and output are the only operators close enough to possibly encode the match operator, where the link names result from the translation of the match variables. But it also reveals the reason why communication is not strong enough. Translated match variables have to be free in the encoding of the match prefix—to allow for a guarding input to receive a value for a match variable—but they also have to be bound—to avoid unintended interactions between the translated match variables of parallel match encodings. The other π-calculus operators cannot simulate this kind of binding.

5.2 Encodings of the Match Prefix in Pi-Like Calculi

As mentioned in the introduction there are some modifications and extensions of the π-calculus that allow for the encoding of the match prefix. We briefly discuss four different approaches and their relation to our separation result.

In [1] the input prefix $x(z)$ of the π-calculus is replaced by a selective input $x(z \in V)$. A term guarded by $x(z \in V)$ and a term guarded by a matching output prefix $\overline{x}\langle y \rangle$ can communicate (if they are composed in parallel and) only if the transmitted value y is contained in the set V of names specified in the selective input prefix. Accordingly selective input can be used as a conditional guard. As pointed out in [1], selective input allows to encode a match prefix $[a = b]P$ simply by $(\nu x)\,(\overline{x}\langle a \rangle \mid x(y \in \{b\}).[\![P]\!])$, where $[\![P]\!]$ is the encoding of P. Here the test for equality $a = b$ is transferred into the test $a \in \{b\}$. Thus it is not necessary to translate the match variables into communication channels, which allows for this simple encoding.

Mobile ambients [3] extend the asynchronous π-calculus with ambients $n[\,]$, i.e. sides or locations, that (a) can contain processes and other ambients, (b) can be composed in parallel to other ambients and processes, and (c) whose name can be restricted to forbid interaction with its environment. Moreover there are three additional actions prefixes: (1) $\mathsf{in}\,n$ allows an ambient to enter another ambient named n by the rule $m[\mathsf{in}\,n.P \mid Q] \mid n[R] \longmapsto n[m[P \mid Q] \mid R]$, (2) $\mathsf{out}\,n$ allows an ambient to exit its own parent named n by the rule $n[m[\mathsf{out}\,n.P \mid Q] \mid R] \longmapsto m[P \mid Q] \mid n[R]$, and (3) $\mathsf{open}\,n$ dissolves an ambient with name n by the rule $\mathsf{open}\,n.P \mid n[Q] \longmapsto P \mid Q$. As a consequence, communication steps become locale, i.e. can occur only if both communication partners are located in parallel within the same ambient. Hence channel names become superfluous, since communications on different channels can be simulated by communications within different ambients. So the π-input $x(z).P$ is replaced by $(z).P$ and the asynchronous output $\overline{x}\langle y \rangle$ is replaced by $\langle y \rangle$. As pointed out in [17], mobile ambients can encode the match prefix. They suggest to encode a match prefix $[a = b]P$ by the term $M = (\nu xy)\,(x[\mathsf{open}\,a.y[\mathsf{out}\,x] \mid b[\,]] \mid \mathsf{open}\,y.\mathsf{open}\,x.[\![P]\!])$, where $[\![P]\!]$ is the encoding of P. Since there are no channel names, the match variables are translated into the new capabilities of mobile ambients, namely into $\mathsf{open}\,a$ and an ambient with name b. $\mathsf{open}\,a$ can only be reduced if $a = b$, i.e. if either $a = b$ holds from the beginning or if a and b are unified by a substitution induced by a surrounding input, as e.g. in $(b).M \mid \langle a \rangle$. Note that, to enable this substitution, the match variables a and b have—as shown in our proof above—to be translated into free names. Here the ambient x and its restriction ensure that there are no unintended interactions between the translated match variables of parallel match encodings. More precisely the ambient x encapsulates the translation of the test for equality $a = b$ and the restriction (νx) ensures that the environment cannot interfere, i.e. no other action on the names a or b can reduce the $\mathsf{open}\,a$ or can target the ambient b inside of x, because the restriction forbids other processes to enter x. So in mobile ambients it is not necessary to translate the match variables into bound names, which allows for the encoding.

[20] extend the pi-calculus with an additional operator $P \setminus z$ called blocking. Blocking forbids for P to perform a visible action with the blocked name z as subject or bound object. By [20] this allows to encode a match prefix $[a = b]P$ by the term $(\nu w)\,((\overline{a}\langle y \rangle.0 \mid b(z).\overline{w}\langle y \rangle.0) \setminus a \setminus b \mid w(z).[\![P]\!])$, where $[\![P]\!]$ is the encoding of P and $z \notin \mathbf{fn}(P)$. As suggested by our proof above, the match prefix is translated into a communication and the match variables are translated into the channel names of the respective communication partners. To communicate the channel names have to be equal, i.e. again either $a = b$ holds from the beginning or a and b have to be unified by a substitution induced by a surrounding input. To enable such a substitution, the match variables a and b have—as shown in our proof above—to be translated into free names. Here the new blocking operator ensures that there are no unintended interactions between the translated match variables of parallel match encodings. More precisely $M \setminus a \setminus b$ ensures that M cannot interact with another term over a or b—thus blocking behaves as a binding operator w.r.t. reduction steps—but blocking does not bind the names a and b such that they can be affected by substitution. Thus our proof explicitly reveals the features that due to [20] allow to encode the match

prefix by means of blocking.

[2] extends the π-calculus by so-called polyadic synchronisation, i.e. instead of single names as in the π-calculus channel names can be constructed by combining several names. Thus e.g. in the variant of the π-calculus with polyadic synchronisation, where each channel name consists of exactly two names, the input prefix becomes $x_1 \cdot x_2(z)$ and the (matching) output prefix becomes $\overline{x_1 \cdot x_2}\langle y \rangle$. An input and an output guarded term (that are composed in parallel) can communicate if the composed channel names are equal. By [2] this extension allows to encode the match prefix. They suggest to translate $[a = b]\, P$ by $(\nu x)\left(\overline{x \cdot b}\langle y \rangle \mid x \cdot a(z).[\![P]\!]\right)$, where $[\![P]\!]$ is the encoding of P and $x, z \notin \mathbf{fn}(P)$. Again, as suggested by our proof above, the match prefix is translated into a communication and the match variables are translated into (parts of) the channel names of the respective communication partners. But polyadic synchronisation allows to combine the free match variables—used to allow for a guarding input to receive a value—and the bound name x—used to avoid unintended interactions between the translated match variables of parallel match encodings—within a single communication channel. Again our proof explicitly reveals the features that due to [2] allow to encode the match prefix by means of polyadic synchronisation.

6 Conclusions

We provide a novel separation result showing that there is no valid encoding from the full π-calculus into its variant without the match prefix. In contrast to the former approach in [2] we strengthen the result in two ways:

1. We considerably weaken the set of requirements, in particular with respect to the criterion that is called reasonable semantics in [2]. Instead, we use the framework of criteria designed by Gorla for language comparison.
2. The so obtained proof reflects our intuition on the match prefix and reveals the problem that prevents its encoding. A valid encoding of the match prefix would need to translate the prefix into a (set of) communication step(s) on links that result from the translation of the match variables. These links have to be free—to allow for a guarding input to receive a value for a match variable—but they also have to be bound—to avoid unintended interactions between parallel match encodings. This kind of binding cannot be simulated by a π-calculus operator different from the match prefix.

This further underpins that the match prefix cannot be derived in the π-calculus.

In Section 5.2 we discuss four modifications and extensions of the π-calculus that allow to encode the match prefix. In the first encoding approach the match prefix is replaced by another (more general) conditional guard. But the other approaches use extensions or modifications of the π-calculus to encode the match prefix by using features that allow to circumvent the binding problem in the encoding of the match prefix that is pointed out in our proof. Thus further works can use the here presented explicit formulation of the reason, that forbids for encodings of the match prefix in the π-calculus, to encode the match prefix in other calculi.

References

[1] C. Bodei, P. Degano & C. Priami (2005): *Checking security policies through an enhanced Control Flow Analysis*. *Journal of Computer Science* 13(1), pp. 49–85.

[2] M. Carbone & S. Maffeis (2003): *On the Expressive Power of Polyadic Synchronisation in the π-calculus*. *Nordic Journal of Computing* 10(2), pp. 70–98.

[3] L. Cardelli & A.D. Gordon (2000): *Mobile ambients*. *Theoretical Computer Science* 240(1), pp. 177–213, doi:10.1016/S0304-3975(99)00231-5.

[4] M. Giunti (2013): *Algorithmic type checking for a pi-calculus with name matching and session types*. *The Journal of Logic and Algebraic Programming* 82(8), pp. 263–281, doi:10.1016/j.jlap.2013.05.003.

[5] D. Gorla (2010): *A taxonomy of process calculi for distribution and mobility*. *Distributed Computing* 23(4), pp. 273–299, doi:10.1007/s00446-010-0120-6.

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

[7] D. Gorla & U. Nestmann (2014): *Full Abstraction for Expressiveness: History, Myths and Facts*. *Mathematical Structures in Computer Science*. To appear.

[8] R. Milner (1999): *Communicating and Mobile Systems: The π-Calculus*. Cambridge University Press.

[9] R. Milner, J. Parrow & D. Walker (1992): *A Calculus of Mobile Processes, Part I and II*. *Information and Computation* 100(1), pp. 1–77, doi:10.1016/0890-5401(92)90008-4, 10.1016/0890-5401(92)90009-5.

[10] R. Milner & D. Sangiorgi (1992): *Barbed Bisimulation*. In: *Proceedings of ICALP*, *LNCS* 623, Springer, pp. 685–695, doi:10.1007/3-540-55719-9_114.

[11] C. Palamidessi (2003): *Comparing the Expressive Power of the Synchronous and the Asynchronous π-calculus*. *Mathematical Structures in Computer Science* 13(5), pp. 685–719, doi:10.1145/263699.263731.

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

[13] K. Peters (2012): *Translational Expressiveness*. PhD, Technische Universität Berlin.

[14] K. Peters & U. Nestmann (2014): *Breaking Symmetries*. *To Appear in Mathematical Structures in Computer Science*, doi:10.4204/EPTCS.41.10.

[15] K. Peters, U. Nestmann & U. Goltz (2013): *On Distributability in Process Calculi*. In: *Proceedings of ESOP*, *LNCS* 7792, Springer, pp. 310–329, doi:10.1007/978-3-642-37036-6_18.

[16] K. Peters, T. Yonova-Karbe & U. Nestmann (2014): *Matching in the Pi-Calculus (Technical Report)*. Technical Report, TU Berlin. Available at arXiv.org.

[17] I.C.C. Phillips & M.G. Vigliotti (2004): *Electoral Systems in Ambient Calculi*. In: *Proceedings of FoSSaCS*, *LNCS* 2987, pp. 408–422, doi:10.1007/978-3-540-24727-2_29.

[18] D. Sangiorgi (1996): *A theory of bisimulation for the π-calculus*. *Acta Informatica* 33(1), pp. 69–97, doi:10.1007/s002360050036.

[19] D. Sangiorgi & D. Walker (2001): *The π-calculus: A Theory of Mobile Processes*. Cambridge University Press.

[20] J.L.F. Vivas (2001): *Dynamic Binding of Names in Calculi for Mobile Processes*. PhD, Royal Institute of Technology, Sweden.