Breaking symmetries

Kirstin Peters, Uwe Nestmann

Angaben zur Veröffentlichung / Publication details:

Peters, Kirstin, and Uwe Nestmann. 2010. "Breaking symmetries." *Electronic Proceedings in Theoretical Computer Science* 41: 136–50. https://doi.org/10.4204/eptcs.41.10.

Breaking Symmetries

Kirstin Peters
Technische Universität Berlin, Germany
kirstin.peters@tu-berlin.de

Uwe Nestmann
Technische Universität Berlin, Germany
uwe.nestmann@tu-berlin.de

A well-known result by Palamidessi tells us that π_{mix} (the π-calculus with mixed choice) is more expressive than π_{sep} (its subset with only separate choice). The proof of this result argues with their different expressive power concerning leader election in symmetric networks. Later on, Gorla offered an arguably simpler proof that, instead of leader election in symmetric networks, employed the reducibility of "incestual" processes (mixed choices that include both enabled senders and receivers for the same channel) when running two copies in parallel. In both proofs, the role of *breaking (initial) symmetries* is more or less apparent. In this paper, we shed more light on this role by re-proving the above result—based on a proper formalization of what it means to break symmetries—without referring to another layer of the distinguishing problem domain of leader election.

Both Palamidessi and Gorla rephrased their results by stating that there is no uniform and reasonable encoding from π_{mix} into π_{sep}. We indicate how the respective proofs can be adapted and exhibit the consequences of varying notions of uniformity and reasonableness. In each case, the ability to break initial symmetries turns out to be essential.

1 Introduction

Palamidessi's well-known result [Pal03] tells us that π_{mix} (the π-calculus with mixed choice) is more expressive than π_{sep} (its subset with only separate choice). More technically, the result states that there exists no "good"—i.e., uniform (structure-preserving) and reasonable (semantics-preserving)—encoding from π_{mix} into π_{sep}. Nestmann [Nes00] proved that there is a "good" encoding from π_{sep} to π_{a} (the choice-free asynchronous subset of the π-calculus). He also exhibited various encodings from π_{mix} to π_{sep}, which were not considered "good" by Palamidessi, as they were not uniform or reasonable enough.

Palamidessi's proof [Pal03] argues with the different expressive power of the involved calculi concerning leader election in symmetric networks. More precisely, Palamidessi proves that there is no symmetric network in π_{sep} that solves leader election, whereas there are such networks in π_{mix}. The proof implicitly uses the fact that it is not possible in π_{sep} to break initial symmetries, while this is possible in π_{mix}. To this end, a rather strong notion of symmetry consisting of a syntactic and a semantic component is used to ensure that solving leader election requires breaking initial symmetries. With this result, inspired by Bougé's work [Bou88] in the context of *CSP*, Palamidessi proves that there is no uniform and reasonable encoding from π_{mix} into π_{sep}.

Later on, Gorla [Gor08b] offered an arguably simpler proof for the non-existence of a "good" encoding from π_{mix} into π_{sep}. Instead of leader election in symmetric networks, it employed the reducibility of "incestual" processes (mixed choices that include both enabled senders and receivers for the same channel) when running two copies in parallel. Gorla's proof does not explicitly use a notion of symmetry.

Palamidessi's proof that there are no symmetric networks in π_{sep} that solve leader election addresses the *absolute* expressive power of π_{sep}, whereas the proofs of the non-existence of a uniform encoding by Palamidessi and Gorla address the often-called *relative* expressive power of the languages [Par08]. In the following, we discuss these two approaches in more detail, as this allows us to clarify the role of symmetry-breaking in the respective proofs.

S. Fröschle, F.D. Valencia (Eds.): Workshop on
Expressiveness in Concurrency 2010 (EXPRESS'10).
EPTCS 41, 2010, pp. 136–150, doi:10.4204/EPTCS.41.10

The *absolute expressive power* of a language describes what kind of behaviour or operations on behaviour are expressible in it (see [Par08, Gor08a, Gor08b]). Analysing the absolute expressive power of a language usually consists of analysing which "problems" can be solved in it and which can not. It is often difficult to identify a suitable problem instance or problem domain to properly measure the expressive power of a language. For instance, one might consider Turing-completeness to measure the computational power of a language. In fact, Turing-completeness has been used in the context of process algebras, e.g., for Linda [BGZ00]. Instead, Palamidessi, inspired by Bougé [Bou88], uses the distributed coordination problem of leader election. More precisely, the problem refers to initially symmetric networks, where all potential leaders have equal chances and all processes run the same—read: symmetric—code. There, to solve the leader election problem, it is required that in all possible executions a leader is elected. Usually, it is argued that it is necessary—again in all possible executions—to break the initial symmetry in order to do so. On the other hand, if there is just a single execution in which the symmetry is somehow perpetually maintained or at least restored, then also leader election may fail, and thus the leader election problem is not solved. One may conclude that, at a closer look, Palamidessi's proof implicitly addresses another problem: the problem of breaking initial symmetries. Therefore, we suggest to promote "breaking symmetries" from a mere auxiliary proof technique to a proper problem of its own. It turns out that, by doing so, we can significantly weaken the definition of symmetry and at the same time provide a stronger proof applicable to problem instances different from leader election.

Now, to *compare* the absolute expressive power of *two* languages, we may simply choose a problem that can be solved in one language, but not in the other language. Actually, as soon as we compare two languages, it makes sense to use the term *relative expressive power*, as we can now relate the two languages. Unfortunately, the terminology was introduced differently. It has been attributed (see [Par08]) to the comparison of the expressive power of two languages by means of the existence or non-existence of encodings from one language into the other language, subject to various conditions on the encoding.[1] Both Palamidessi and Gorla state results of this kind; they prove that there is no uniform and reasonable encoding from π_{mix} into π_{sep}, for varying interpretations of the conditions uniform and reasonable.

In this paper, we show that the problem of breaking initial symmetries, compared to the problem of leader election, appears to be a more suitable problem instance to separate π_{mix} from π_{sep}. There are two great benefits in proving an absolute separation result instead of a translational one. First, in opposite to translational separation results which are always equipped with the conditions on the encoding, we can formulate a separation result without any pre- or side conditions. Second, as we show in Section 5, we can prove several translational separation results due to different definitions of reasonableness as simple consequences of our absolute separation result. For our work, we had to develop answers to two related questions of definition:

- How exactly should one define *symmetric* networks?
- What exactly does it mean to *break symmetries*?

The main contributions of this paper are then as follows. (1) We present a separation result between π_{mix} and π_{sep} that does not require any additional preconditions. In particular, it is completely independent of what it means for an encoding to be "good" or "reasonable". (2) Since we use a weaker notion of symmetry, and because we do not focus on the leader election problem, our separation result is more

[1] In our opinion, the denotation "relative expressive power" is misleading. First, as mentioned above, also the absolute expressive power can directly be used to *relate* two languages. Second, results on the encodability of a language have to be understood relative to the specific conditions on the encoding—it is not always clear to what aspect the "relative" refers. Thus, in this paper, we prefer the notion of *translational expressive power* to refer to comparisons of the expressiveness of two languages by analysing the existence or non-existence of an encoding, subject to various conditions.

general than the one in [Pal03], i.e., it widens the gap between π_{mix} and π_{sep}. It also allows us to derive a number of translational separation results using counterexamples different from leader election. (3) We prove a stronger translational separation result in comparison to [Pal03, VPP07] and (the first setting of) [Gor08b] by weakening the conditions on the encodings used.

Overview of the Paper. In §2, we introduce the two process calculi that we intend to compare. In §3, we revisit the notion of symmetry used by Palamidessi to propose her separation result and define symmetry as we use it. In §4, we prove the separation result, i.e., we prove that π_{mix} is strictly more expressive as π_{sep}, by proving the inability of π_{sep} to break initial symmetries. Based on this result, we prove in §5 that there is no uniform and reasonable encoding from π_{mix} to π_{sep} examining different notions of reasonableness. We conclude with §6.

2 Technical Preliminaries

In the following, let $\mathcal{N}$ denote a countable set of names. As is common nowadays, we present the π-calculus including mixed guarded choice, but without match or mismatch operator [SW01, Pal03].

Definition 2.1 (π-calculus). The processes of the *π-calculus*, denoted by $\mathcal{P}_{\text{mix}}$, are given by

$$P \quad ::= \quad \sum_i \alpha_i.P_i \quad | \quad P \mid P \quad | \quad (\nu z)\, P \quad | \quad !P \quad \text{, where}$$
$$\alpha \quad ::= \quad \overline{x}y \quad | \quad x(z) \quad | \quad \tau$$

(End of Definition 2.1)

Note that the process term $\sum_i \alpha_i.P_i$ represents *finite* guarded choice; as usual, the term $\alpha_1.P_1 + \alpha_2.P_2$ denotes binary choice, and we use $\mathbf{0}$ as abbreviation for the empty sum.

In the π-calculus with separate choice, both output and input can be used as guards, but within a choice term either there are no input or no output guards, i.e., we have input and output guarded choice, but no mixed choice.

Definition 2.2 (π-calculus with separate choice). The processes of the *π-calculus with separate choice*, denoted by $\mathcal{P}_{\text{sep}}$, are given by

$$P \quad ::= \quad \sum_i \alpha_i^I.P_i \quad | \quad \sum_i \alpha_i^O.P_i \quad | \quad P \mid P \quad | \quad (\nu z)\, P \quad | \quad !P \quad \text{, where}$$
$$\alpha^I \quad ::= \quad x(z) \quad | \quad \tau \qquad \text{and} \qquad \alpha^O \quad ::= \quad \overline{x}y \quad | \quad \tau$$

(End of Definition 2.2)

We use $x, x', x_1, \ldots, y, y', y_1, \ldots, z, z', z_1, \ldots$ to range over names and capital letters $P, P', P_1, \ldots, Q, R, \ldots$ to range over processes. We often omit $\mathbf{0}$ in longer terms. If we refer to processes without further requirements we mean elements of $\mathcal{P}_{\text{mix}}$; we sometimes use just $\mathcal{P}$ when the discussion applies to both.

Let $\mathcal{A} \stackrel{\text{def}}{=} \{\, xy,\ \overline{x}y,\ \overline{x}(y) \mid x, y \in \mathcal{N} \,\}$ denote the set of visible actions, where xy denotes *free input*, $\overline{x}y$ denotes *free output* and $\overline{x}(y)$ denotes *bound output*. Let τ denote an internal not visible action. Let $\mathcal{L}$ be the corresponding set of *labels*, i.e., $\mathcal{L} = \mathcal{A} \cup \{\tau\}$. We use $\mu, \mu', \mu_1, \ldots$ to range over labels. Let $\text{fn}(P)$ and $\text{fn}(\mu)$ denote the sets of *free names* in P and μ, respectively. Let $\text{bn}(P)$ and $\text{bn}(\mu)$ denote the sets of *bound names* in P and μ, respectively. Likewise, $\text{n}(P)$ and $\text{n}(\mu)$ denote the sets of all *names* occurring

in P and μ. Their definitions are completely standard. We assume that there are no clashes between free and bound names in terms, i.e., in any term the set of bound and free names are disjoint.

The operational semantics of $\mathscr{P}_{\text{mix}}$ and $\mathscr{P}_{\text{sep}}$ are jointly given by the transition rules in Figure 1, where congruence $\equiv$ is defined (according to [Pal03]) by the following rules:

1. $P \equiv Q$ if Q can be obtained from P by alpha-conversion
2. $(\nu x)\, P \mid Q \equiv (\nu x)\,(P \mid Q)$ if $x \notin \text{fn}(Q)$
3. $P \mid Q \equiv Q \mid P$

$$\textsc{I-Sum}\quad \sum_i \alpha_i.P_i \xrightarrow{xy} \{\, y/z \,\} P_j \quad a_j = x(z) \qquad \textsc{O/}\tau\textsc{-Sum}\quad \sum_i \alpha_i.P_i \xrightarrow{\alpha_j} P_j \quad \alpha_j = \overline{x}y \text{ or } \alpha_j = \tau$$

$$\textsc{Par}\quad \frac{P \xrightarrow{\mu} P'}{P|Q \xrightarrow{\mu} P'|Q} \quad \text{bn}(\mu) \cap \text{fn}(Q) = \emptyset$$

$$\textsc{Comm}\quad \frac{P \xrightarrow{\overline{x}y} P' \quad Q \xrightarrow{xy} Q'}{P|Q \xrightarrow{\tau} P'|Q'} \qquad \textsc{Close}\quad \frac{P \xrightarrow{xy} P' \quad Q \xrightarrow{\overline{x}(y)} Q'}{P \mid Q \xrightarrow{\tau} (\nu y)\,(P' \mid Q')} \quad y \notin \text{fn}(P)$$

$$\textsc{Res}\quad \frac{P \xrightarrow{\mu} P'}{(\nu z)\, P \xrightarrow{\mu} (\nu z)\, P'} \quad z \notin \text{n}(\mu) \qquad \textsc{Rep}\quad \frac{P \mid !P \xrightarrow{\mu} P'}{!P \xrightarrow{\mu} P'}$$

$$\textsc{Open}\quad \frac{P \xrightarrow{\overline{x}y} P'}{(\nu y)\, P \xrightarrow{\overline{x}(y)} P'} \quad x \neq y \qquad \textsc{Cong}\quad \frac{P' \equiv P \quad P \xrightarrow{\mu} Q \quad Q \equiv Q'}{P' \xrightarrow{\mu} Q'}$$

Figure 1: Operational semantics

As usual, the tuple notation $\tilde{x} \in \mathscr{T}(\mathscr{N})$ denotes finite sequences $x_1, \ldots, x_n$ of names in $\mathscr{N}$, i.e., $\mathscr{T}(M)$ denotes the set of tuples over a set M. Moreover, we use $(\nu \tilde{x})$ for a sequence $\tilde{x} = x_1, \ldots, x_n$ to abbreviate $(\nu x_1) \ldots (\nu x_n)$ and $\tilde{x} \setminus M$ for a set of names M to denote the sequence of names $\tilde{x}$ without the occurrences of name y for all $y \in M$. We also use the tuple notation for other kinds of data, like actions or labels.

A *network* is a process $(\nu \tilde{x})\,(P_1 \mid \ldots \mid P_n)$ for some $n \in \mathbb{N}$, $P_1, \ldots, P_n \in \mathscr{P}$ and $\tilde{x} \in \mathscr{T}(\mathscr{N})$. We refer to $P_1, \ldots, P_n$ as the processes of the network.

We use σ, σ', σ_1, ... to range over substitutions. A substitution is a set $\{\, x_1/y_1, \ldots, x_n/y_n \,\}$ of rules to rename free names of a term. $\{\, x_1/y_1, \ldots, x_n/y_n \,\}(P)$ is defined as the result of replacing all occurrences of y_i by x_i for $i \in \{\, 1, \ldots, n \,\}$, possibly applying alpha-conversion to avoid capture or name clashes. For all names $\mathscr{N} \setminus \{\, y_1, \ldots, y_n \,\}$ the substitution behaves as identity function. Let **id** denote identity, i.e., **id** is the empty substitution $\mathbf{id} = \{\}$.

As usual, $P \xrightarrow{\mu} P'$ denotes a step from P to P', where μ is either a label of an action or τ. Moreover let $P \to (P \not\to)$ denote existence (non-existence) of a step from P, i.e., there is (no) $P' \in \mathscr{P}$ and (no) $\mu \in \mathscr{L}$ such that $P \xrightarrow{\mu} P'$. A (partial) execution is a sequence of steps $P \xrightarrow{\mu_1, \ldots, \mu_n} P'$ such that $P \xrightarrow{\mu_1} H_1 \xrightarrow{\mu_2} \ldots \xrightarrow{\mu_{n-1}}$

$H_{n-1} \xrightarrow{\mu_n} P'$ for some $P', H_1, \ldots, H_{n-1} \in \mathscr{P}$ with the sequence $\mu_1, \ldots, \mu_n$ of observable and unobservable actions, i.e., $\mu_1, \ldots, \mu_n \in \mathscr{L}$. Accordingly $P \xrightarrow{\tilde{\mu}} P' \not\rightarrow$ denotes a finite execution from P to P' with the sequence of actions $\tilde{\mu} \in \mathscr{T}(\mathscr{L})$.

3 Semantic versus Syntactic Symmetry

Palamidessi in [Pal03] proved that π_{mix} is strictly more expressive than π_{sep} by proving that the former can solve leader election in symmetric networks while the latter can not. The leader election problem consists of choosing a leader among the processes of a network. In [Pal03], a special channel *out* is assumed to propagate the index of the winning process, i.e., the leader. The leader election problem is solved by a network iff in each of its executions each process propagates the same process index over *out* and no other index is propagated.

As already Bougé did for *CSP* in [Bou88], Palamidessi uses a *semantic* definition of symmetry. Intuitively, the *syntactic component* of the symmetry definition in [Bou88, Pal03, VPP07] states two processes as symmetric iff they are identical modulo some renaming according to a permutation σ on their free names. Bougé [Bou88] argues why a syntactic notion of symmetry does not suffice considering the leader election problem to distinguish $CSP_{\text{i/o}}$, i.e., *CSP* where input and output commands may appear in guards, and CSP_{in}, i.e., *CSP* where only input commands may appear in guards. He presents two networks in CSP_{in} each solving leader election although each should be considered as syntactically symmetric. The following example presents such a syntactically symmetric network solving leader election in π_{sep}:

$$N \triangleq P \mid \sigma(P) \quad \text{with} \quad P = \overline{x} \mid x.\overline{out}\ 1 + y.\overline{out}\ 2 \quad \text{and} \quad \sigma = \{\, {}^{x}/_{y}, {}^{y}/_{x} \,\} \tag{1}$$

N is syntactically symmetric with respect to the permutation σ, i.e., $N = P_1 \mid P_2$ and P_2 is equal to P_1 modulo the exchange of x and y according to σ. Moreover N solves the leader election problem.

To overcome these problems the *semantic component* of the symmetry definition is designed to be strongly connected to the problem considered, i.e., leader election in this case. Intuitively, its purpose is to ensure that the only way to solve the leader election problem is to break the initial symmetry of the given network. Note that N does *not* break the initial syntactic symmetry, because e.g. in the execution $N \xrightarrow{\tau} P \mid \overline{out}\ 1 \xrightarrow{\tau} \overline{out}\ 1 \mid \overline{out}\ 1 \xrightarrow{\overline{out}\ 1} \mathbf{0} \mid \overline{out}\ 1 \xrightarrow{\overline{out}\ 1} \mathbf{0} \mid \mathbf{0} \not\rightarrow$ each second step results in a network that is syntactically symmetric with respect to σ. So, without this semantic part in the definition of symmetry, the leader election problem can not be used to distinguish π_{mix} and π_{sep} (or $CSP_{\text{i/o}}$ and CSP_{in}).

Semantic symmetry. We revisit Palamidessi's notion of symmetry for the π-calculus as of [Pal03]. Note that the involved definitions are based on the ones introduced by Bougé in [Bou88] for *CSP*.

According to [Pal03], a hypergraph is a tuple $H = \langle N, X, t \rangle$, where N and X are finite sets whose elements are called nodes and edges and t, called type, is a function assigning to each edge the set of nodes connected by this edge. An automorphism on a hypergraph is a pair $\sigma = \langle \sigma_N, \sigma_X \rangle$ such that $\sigma_N : N \to N$ and $\sigma_X : X \to X$ are permutations which preserve the type of edges. Given a hypergraph H and σ on H the orbit of a name n is the set of nodes in which the iterations of σ map n.

A network $P \equiv (\nu \tilde{x})\ (P_1 \mid \ldots \mid P_k)$ of k processes solves the leader election problem if for every computation of P there exists an extension of the computation and there exists an index $n \in \{\, 1, \ldots, k \,\}$ such that for each process the extended computation contains one output action of the form $\overline{out}\ n$ and no other action $\overline{out}\ m$ with $m \neq n$. The hypergraph associated to a network P is the hypergraph $H(P) = \langle N, X, t \rangle$

with $N = \{ 1,\ldots,k \}$, $X = \mathrm{fn}(P_1 \mid \ldots \mid P_k) \setminus \{ \mathit{out} \}$, and for each $x \in X$, $t(x) = \{ n \mid x \in \mathrm{fn}(P_n) \}$. Given a network P and the hypergraph $H(P)$ associated to P an automorphism on P is any automorphism $\sigma = \langle \sigma_N, \sigma_X \rangle$ on $H(P)$ such that σ_X coincides with σ_N on $N \cap X$ and σ_X preserves the distinction between free and bound names.

A network P with the associated hypergraph $H(P) = \langle N, X, t \rangle$ and an automorphism σ on P is symmetric with respect to σ iff for each node $i \in N$, $P_{\sigma(i)} \equiv_\alpha \sigma(P_i)$[2], holds where $\equiv_\alpha$ denotes equality modulo alpha conversion.

To distinguish π_{mix} and π_{sep} Palamidessi shows that a network $P \in \mathscr{P}_{\mathrm{sep}}$ which is symmetric with respect to an automorphism σ on P with only one orbit can not solve the leader election problem while this is possible in π_{mix}.

The main point of the semantic component of symmetry is that the special channel *out* can not be renamed by σ while the indices of the processes of the network must be permuted by σ. With that, the network N in (1) above is *not* symmetric according to [Pal03]. This allows Palamidessi to prove that for each execution of a network in $\mathscr{P}_{\mathrm{sep}}$, which is symmetric with respect to an automorphism σ, whenever there is an output $\overline{\mathit{out}}\, i$ there is an output $\overline{\mathit{out}}\, \sigma(i)$ with $\sigma(i) \neq i$ as well, which contradicts the leader election problem. This explains why in [Bou88, Pal03, VPP07] such an effort is spent to define symmetry.

Nevertheless it turns out that we do not need the leader election problem to distinguish π_{mix} and π_{sep}. The main argument in the proof of [Pal03] that there is no symmetric network in $\mathscr{P}_{\mathrm{sep}}$ solving leader election is that it is impossible in π_{sep} to break symmetries.

Syntactic symmetry. As mentioned in the introduction, we directly focus on the problem of breaking symmetries instead of concentrating on leader election. Thus, we can release most of the above conditions for symmetry. Moreover, we abandon the notion of hypergraphs and automorphisms. Instead, we use a simple syntactic definition of symmetry that, as mentioned above, states two processes as symmetric iff they are identical modulo some renaming according to a permutation σ on their free names.

Definition 3.1 (Symmetry relation). A *symmetry relation of degree n* is a permutation $\sigma : \mathscr{N} \to \mathscr{N}$, such that $\sigma^n = \mathbf{id}$.

Let $\mathrm{Sym}(n, \mathscr{N})$ denote the set of symmetry relations of degree n over $\mathscr{N}$ and let $\sigma^0 = \mathbf{id}$. (End of Definition 3.1)

Note that this definition does not require that n is the minimal degree of σ; consequently, the condition that σ is an automorphism with only one orbit is released. A symmetric network is then a network of n processes that are equal except for some renaming according to a symmetry relation σ.

Definition 3.2 (Symmetric network). Let $P \in \mathscr{P}$. Let sequence $\tilde{x}$ contain only free names of P. Let $n \in \mathbb{N}$. Let σ be a symmetry relation of degree n over $\mathscr{N} \setminus \mathrm{bn}(P)$. Let $\tilde{x}$ be closed under σ. Then

$$\left[P\right]_{\sigma}^{n,\tilde{x}} = (\nu \tilde{x}) \left(\sigma^0(P) \mid \ldots \mid \sigma^{n-1}(P) \right)$$

is a *symmetric network of degree n*. (End of Definition 3.2)

Note that, in the following proofs, we make use of the fact that names bound in P are bound in each other process of $\left[P\right]_{\sigma}^{n,\tilde{x}}$ as well, so we explicitly forbid alpha-conversion here. In the following, whenever we assume some symmetric network $\left[P\right]_{\sigma}^{n,\tilde{x}}$, we implicitly assume the respectively quantified parameters: a

[2] In [Bou88] and [VPP07] formally slightly different conditions but with the same effect are used.

process $P \in \mathscr{P}$, a sequence $\tilde{x}$ containing only free names of P, a network size $n \in \mathbb{N}$, a symmetry relation σ of degree n over $\mathscr{N} \setminus \mathrm{bn}(P)$.

The main difference of our definition to the definition of a symmetric network in [Pal03] is that, in [Pal03], the processes of a symmetric network are numbered consecutively and for each process P_i within the symmetric network $P_{\sigma(i)} \equiv \sigma(P_i)$ holds. Thus, each symmetric network in [Pal03] is a symmetric network for our definition, but not vice versa. Our definition of symmetry is weaker.

We use an index-guided form of substitution to replace single processes within a symmetric network.

Definition 3.3 (Indexed substitution). Let $\left[P\right]_{\sigma}^{n,\tilde{x}}$ be a symmetric network. An *indexed substitution* of some processes within a symmetric network, denoted by $\{\, i_1 \mapsto Q_1, \ldots, i_m \mapsto Q_m \,\} \left[P\right]_{\sigma}^{n,\tilde{x}}$ for some processes $Q_1, \ldots, Q_m \in \mathscr{P}$ and $i_1, \ldots, i_m \in \{\, 0, \ldots, n-1 \,\}$ such that for all $j,k \in \{\, 1, \ldots, m \,\}$ $j \neq k$ implies $i_j \neq i_k$, is the result of exchanging $\sigma^{i_k}(P)$ in $\left[P\right]_{\sigma}^{n,\tilde{x}}$ by Q_k for all $k \in \{\, 1, \ldots, m \,\}$. (End of Definition 3.3)

Obviously $\{\, i_1 \mapsto Q_1, \ldots, i_m \mapsto Q_m \,\} \left[P\right]_{\sigma}^{n,\tilde{x}}$ is a network; in general, however, it is not symmetric with respect to σ.

4 Symmetric Executions

We explicitly prove that in π_{sep} it is not possible to break initial symmetries, i.e., starting with a symmetric network there is always at least one execution preserving the symmetry. We refer to such an execution as *symmetric execution*. Let us consider a symmetric network $\left[P\right]_{\sigma}^{n,\tilde{x}}$ of degree n. Of course, if only one process does a step on its own, then all the other processes of the network can mimic this step and thus restore symmetry. So, there is a symmetry preserving execution if there is no communication between the processes of the network. The most interesting case is how the symmetry is restored after a communication between two processes of the network has temporarily destroyed it. Both cases are reflected in the proof of Theorem 4.4.

Apart from symmetric networks, we use the notion of a symmetric sequence of actions. Similarly to symmetric networks, in which a symmetry relation is applied to processes to derive symmetric processes, a symmetric sequence of actions is the result of applying a symmetry relation to action labels. It is sometimes necessary to translate a bound output to an according unbound output because a network can send a bound name several times but only the first of this outputs will be bound.

Definition 4.1 (Symmetric sequence of actions). Let $\mu \in \mathscr{L}$ be an action label, let $\tilde{x} \in \mathscr{T}(\mathscr{N})$ be a sequence of names and σ a symmetry relation of degree $n \in \mathbb{N}$. Then $[\mu]_{\sigma}^{n,\tilde{x}}$ denotes the sequence $\mu_1, \ldots, \mu_n$ of n labels such that $\mu_1, \ldots, \mu_n \in \mathscr{L}$, $\mu_1 = \mu$ and for $i \in \{\, 2, \ldots, n \,\}$:

$$\mu_i = \begin{cases} \tau, & \text{if } \mu = \tau \\ \sigma^i(a)\, b, & \text{if } \mu = ab \\ \overline{\sigma^i(a)}\,\sigma^i(b), & \text{if } \mu = \overline{a}b \text{ or } \left(\mu = \overline{a}(b) \text{ and } \sigma^i(b) \notin \tilde{x} \setminus \{\, b, \sigma(b), \ldots, \sigma^{i-1}(b) \,\}\right) \\ \overline{\sigma^i(a)}\left(\sigma^i(b)\right), & \text{if } \mu = \overline{a}(b) \text{ and } \sigma^i(b) \in \tilde{x} \setminus \{\, b, \sigma(b), \ldots, \sigma^{i-1}(b) \,\} \end{cases}$$

Sometimes we refer to $\mu_2, \ldots, \mu_n$ as the symmetric counterparts of μ. (End of Definition 4.1)

Intuitively, a symmetric execution is an execution starting from a symmetric network returning to a symmetric network after any n'th step, and which is either infinite or terminates in a symmetric network. Thereby, each sequence of n steps is labelled by a symmetric sequence of actions.

Definition 4.2 (Symmetric execution). A *symmetric execution* is either a finite execution of length $m \cdot n \in \mathbb{N}$

$$\left[P\right]_{\sigma}^{n,\tilde{x}} \xrightarrow{[\mu_1]_{\sigma_1}^{n,\tilde{x}}} \left[P_1\right]_{\sigma_1}^{n,\tilde{x}_1} \xrightarrow{[\mu_2]_{\sigma_2}^{n,\tilde{x}_1}} \ldots \xrightarrow{[\mu_m]_{\sigma_m}^{n,\tilde{x}_{m-1}}} \left[P_m\right]_{\sigma_m}^{n,\tilde{x}_m} \not\to$$

for some $P_1,\ldots,P_m \in \mathscr{P}$, $\mu_1,\ldots,\mu_m \in \mathscr{L}$, $\tilde{x}_1,\ldots,\tilde{x}_m \in \mathscr{T}(\mathscr{N})$ and $\sigma_1,\ldots,\sigma_m \in \mathrm{Sym}(n,\mathscr{N})$ such that $\sigma \subseteq \sigma_1 \subseteq \ldots \subseteq \sigma_m$ or an infinite execution

$$\left[P\right]_{\sigma}^{n,\tilde{x}} \xrightarrow{[\mu_1]_{\sigma_1}^{n,\tilde{x}}} \left[P_1\right]_{\sigma_1}^{n,\tilde{x}_1} \xrightarrow{[\mu_2]_{\sigma_2}^{n,\tilde{x}_1}} \left[P_2\right]_{\sigma_2}^{n,\tilde{x}_2} \xrightarrow{[\mu_3]_{\sigma_3}^{n,\tilde{x}_2}} \ldots.$$

for some $P_1,P_2,\ldots \in \mathscr{P}$, $\mu_1,\mu_2,\ldots \in \mathscr{L}$, $\tilde{x}_1,\tilde{x}_2,\ldots \in \mathscr{T}(\mathscr{N})$ and $\sigma_1,\sigma_2,\ldots \in \mathrm{Sym}(n,\mathscr{N})$ such that $\sigma \subseteq \sigma_1 \subseteq \sigma_2 \subseteq \ldots$. (End of Definition 4.2)

Note that because of $\sigma \subseteq \sigma_1 \subseteq \ldots$ the symmetry relation can only increase during a symmetric execution such that existing symmetries are preserved. Moreover—as shown in Lemma 4.5—the symmetry relation does only grow in the presence of bound output to capture the renaming done by alpha-conversion. In the absence of bound output we have $\sigma = \sigma_1 = \ldots = \sigma_m$ and $\sigma = \sigma_1 = \sigma_2 = \ldots$ respectively.

Palamidessi proved that π_{sep} enjoys a certain kind of *confluence* property [Pal03]. Let $\overline{x}[y]$ denote an output action, i.e., $\overline{x}[y]$ is either a bound output $\overline{x}(y)$ or an unbound output $\overline{x}y$.

Lemma 4.3. *Let $P \in \mathscr{P}_{\mathrm{sep}}$ be a process. If P can make two steps $P \xrightarrow{\overline{x}[y]} Q$ and $P \xrightarrow{zw} R$ then there exists S such that $Q \xrightarrow{zw} S$ and $R \xrightarrow{\overline{x}[y]} S$.*

Proof of Lemma 4.3. See proof of Lemma 4.1 in [Pal03] at pages 17 to 18. □

With this property we prove that it is not possible to break symmetries in π_{sep}. Intuitively, we show that there is at least one symmetric execution by proving that whenever there is a step destroying symmetry we can restore it in $n-1$ more steps mimicking the first step. The respective existence relies on the standard Lemma in process calculi like the π-calculus that transitions are preserved under substitution. As conclusion it is not possible in π_{sep} to break an initial symmetry in all executions.

Theorem 4.4 (Symmetric Execution). *Every symmetric network in $\mathscr{P}_{\mathrm{sep}}$ has at least one symmetric execution.*

Proof of Theorem 4.4. We first prove the following statement:

Lemma 4.5.

$$\begin{gathered}\forall n \in \mathbb{N} \,.\, \forall \tilde{x} \in \mathscr{T}(\mathscr{N}) \,.\, \forall P \in \mathscr{P}_{\mathrm{sep}} \,.\, \forall \sigma \in \mathrm{Sym}(n, \mathscr{N} \setminus \mathrm{bn}(P)) \,.\, \forall \mu \in \mathscr{L} \,. \\ \left[P\right]_{\sigma}^{n,\tilde{x}} \xrightarrow{\mu} \widehat{P} \text{ implies } \exists P' \in \mathscr{P}_{\mathrm{sep}} \,.\, \exists \tilde{x}' \in \mathscr{T}(\mathscr{N}) \,.\, \exists \mu_2,\ldots,\mu_n \in \mathscr{L} \,.\, \exists \sigma' \in \mathrm{Sym}(n,\mathscr{N}) \,. \\ \widehat{P} \xrightarrow{\mu_2,\ldots,\mu_n} \left[P'\right]_{\sigma'}^{n,\tilde{x}'} \text{ and } \mu,\mu_2,\ldots,\mu_n = [\mu]_{\sigma'}^{n,\tilde{x}} \text{ and } \sigma \subseteq \sigma'\end{gathered}$$

Intuitively it states that given an arbitrary symmetric network $\left[P\right]_{\sigma}^{n,\tilde{x}}$ in $\mathscr{P}_{\mathrm{sep}}$, whenever $\left[P\right]_{\sigma}^{n,\tilde{x}}$ can perform a step then there are exactly $n-1$ more steps that restore symmetry, i.e., that lead to a symmetric network again and the corresponding n steps are labelled by a sequence of symmetric actions. Note that the main line of argumentation of this Lemma is very similar to the proof of Theorem 4.2 in [Pal03] at pages 18 to 23, although we prove a completely different statement. Nevertheless due to the different proof statements the proofs differ in technical details. We only present an informal proof outline here. A more formal proof can be found in [PN10].

Proof outline of Lemma 4.5. $\left[P\right]_{\sigma}^{n,\tilde{x}} \xrightarrow{\mu} \widehat{P}$ can be the result of either an internal μ-step of one process of the network or of a communication between two processes of the network. In the first case, only one process performs a step and the rest of the network remains equal, i.e.:

$$\exists i \in \{\, 0,\ldots,n-1 \,\} \,.\, \exists H \in \mathscr{P}_{\text{sep}} \,.\, \exists \tilde{x}_1 \in \mathscr{T}\left(\mathscr{N}\right) \,.\, \sigma^i\left(P\right) \xrightarrow{\mu} H \text{ and } \widehat{P} \equiv \{\, i \mapsto H \,\} \left[P\right]_{\sigma}^{n,\tilde{x}_1}$$

In this case, we can simply mimic the step of the first process by performing the action according to the $j{+}1$'th label in $\left[\mu\right]_{\sigma'}^{n,\tilde{x}}$ by process $\sigma^{i+j}\left(P\right)$ for each $j \in \{\, 1,\ldots,n-1 \,\}$. By symmetry, each process can perform this step and each step results in a process symmetric to the one produced by the first step such that the n steps lead to a symmetric network again. Difficulties arise only in the case that μ is a bound output. Otherwise, we can choose $\tilde{x}' = \tilde{x}$ and $\sigma' = \sigma$. If μ is a bound output of a name bound in the whole network, we have to reduce $\tilde{x}$ by all names sent by bound outputs in $\left[\mu\right]_{\sigma}^{n,\tilde{x}}$ to obtain $\tilde{x}'$. Note that some outputs in $\left[\mu\right]_{\sigma}^{n,\tilde{x}}$ may be unbound. In this case, we can choose $\sigma' = \sigma$ again. Otherwise, if μ is a bound output of a name bound in a process of the network then, by symmetry, this name is bound in any other process of the network, too. So performing the first step requires alpha-conversion to avoid name capture. To keep track of the names changed by alpha-conversion we have to update σ in this case such that σ' is the union of σ and a permutation on the bound names due to the performed alpha-conversion. In this case, $\tilde{x}' = \tilde{x}$.

In the second case, $\mu = \tau$ and two processes of the network change, i.e.:

$$\begin{aligned}\exists i,j \in \{\, 0,\ldots,n-1 \,\} \,.\, \exists H_1,H_2 \in \mathscr{P}_{\text{sep}} \,.\, \exists z,z' \in \mathscr{N} \,.\, i \neq j \text{ and } \Big(&\sigma^i\left(P\right) \mid \sigma^j\left(P\right) \xrightarrow{\tau} H_1 \mid H_2 \\ &\text{or } \sigma^i\left(P\right) \mid \sigma^j\left(P\right) \xrightarrow{\tau} \left(\nu z,z'\right)\left(H_1 \mid H_2\right)\Big) \text{ and } \widehat{P} \equiv \{\, i \mapsto H_1, j \mapsto H_2 \,\} \left[P\right]_{\sigma'}^{n,\tilde{x}'}\end{aligned}$$

This case is a little bit more difficult, but again with the help of the confluence lemma and the symmetry of the network, we can show that there exists $n{-}1$ steps mimicking the first communication step such that each process is exactly once a sender and once a receiver. Symmetry ensures that each process can perform a sending and a receiving action symmetric to the actions performed in the first step. By the confluence lemma, these two steps can be performed by each process consecutively in an arbitrary order, so each process can first perform the corresponding sending action and afterwards the corresponding receiving action or the other way around. By symmetry, these n steps result in a symmetric network. Again, a bound output in the first step leads to some difficulties. Otherwise, we can choose $\tilde{x}' = \tilde{x}$ and $\sigma' = \sigma$ again. If the first step contains a bound output, then the corresponding name was bound in a process of the network (not in the whole network) and so, by symmetry, it is bound in each process of the network. With that again, we have to perform alpha-conversion. Moreover, the name formerly bound and its renamings due to alpha-conversion are bound in the whole network after the n steps such that we have to update $\tilde{x}$ and σ according to this alpha-conversion to obtain $\tilde{x}'$ and σ'. □

With Lemma 4.5, we can now construct the symmetric execution. We start with an arbitrary symmetric network $\left[P\right]_{\sigma}^{n,\tilde{x}}$. If $\left[P\right]_{\sigma}^{n,\tilde{x}} \not\rightarrow$ we have a symmetric execution of length 0. Otherwise, if $\left[P\right]_{\sigma}^{n,\tilde{x}}$ can perform a step labelled by μ_1 by Lemma 4.5 we can perform $n{-}1$ more steps such that $\left[P\right]_{\sigma}^{n,\tilde{x}} \xrightarrow{\left[\mu_1\right]_{\sigma_1}^{n,\tilde{x}}} \left[P_1\right]_{\sigma_1}^{n,\tilde{x}_1}$. Now we can proceed alike with $\left[P_1\right]_{\sigma_1}^{n,\tilde{x}_1}$ and result either in a finite symmetric execution of length n or we have $\left[P\right]_{\sigma}^{n,\tilde{x}} \xrightarrow{\left[\mu_1\right]_{\sigma_1}^{n,\tilde{x}}} \left[P_1\right]_{\sigma_1}^{n,\tilde{x}_1} \xrightarrow{\left[\mu_2\right]_{\sigma_2}^{n,\tilde{x}_1}} \left[P_2\right]_{\sigma_2}^{n,\tilde{x}_2}$. By recursively repeating this argument, we either get a finite or an infinite symmetric execution. □

Breaking Symmetries. Note that Theorem 4.4 does not state anything about encodability and it does not need a notion of reasonableness either. Instead, it just states without any precondition that every symmetric network in $\mathscr{P}_{\text{sep}}$ has at least one symmetric execution. In contrast, there are symmetric networks in $\mathscr{P}_{\text{mix}}$ without such a symmetric execution, as the following example shows. Consider the network

$$(\nu x,y)\ (P \mid \sigma(P)) \quad \text{with} \quad P = \overline{x}.\overline{1} + y.\overline{2} \quad \text{and} \quad \sigma = \{\, {}^{x}/_{y}, {}^{y}/_{x}, {}^{1}/_{2}, {}^{2}/_{1} \,\}$$

with $\sigma^2 = \mathbf{id}$, i.e., $(\nu x,y)\ (P \mid \sigma(P))$ is a symmetric network in $\mathscr{P}_{\text{mix}}$. It has, modulo structural congruence, exactly the two following executions

$$(\nu x,y)\ (P \mid \sigma(P)) \xrightarrow{\tau} \overline{1} \mid \overline{1} \xrightarrow{\overline{1}} \overline{1} \xrightarrow{\overline{1}} \mathbf{0}$$
$$(\nu x,y)\ (P \mid \sigma(P)) \xrightarrow{\tau} \overline{2} \mid \overline{2} \xrightarrow{\overline{2}} \overline{2} \xrightarrow{\overline{2}} \mathbf{0}$$

and even none of them is symmetric; the initial symmetry is broken. So Theorem 4.4 proves a difference in the absolute expressive power between π_{sep} and π_{mix}[3].

Fact 4.6. The full π-calculus is strictly more expressive as the π-calculus without mixed choice.

5 Non-Existence of Uniform Encodings

As done by Palamidessi [Pal03] and also by Gorla [Gor08b], we now also prove that there is no uniform and reasonable encoding from π_{mix} into π_{sep}, but here using Theorem 4.4 which states a difference in the absolute expressive power of the two calculi. It is no real surprise that this absolute result leads to differences in the translational expressiveness of the languages. Because uniform encodings preserve symmetries—or at least enough of the symmetric nature of the terms—, the non-existence of a uniform and reasonable encoding is a natural consequence of the difference in their absolute expressiveness. Unfortunately, there is no agreement on the minimal requirements of a reasonable encoding, so we can not formally prove this result in general, although we believe that it holds for any meaningful Definition of reasonableness. Instead to underpin our assertion we prove it in the settings of [Pal03] and [Gor08b].

According to [Pal03], an encoding is uniform if it translates the parallel operator homomorphically and preserves renamings, i.e., for all permutations of names σ there exists a permutation of names θ such that $[\![\, \sigma(P) \,]\!] = \theta([\![\, P \,]\!])$. Vigliotti et al. [VPP07] additionally require that the permutations σ and θ are compatible on observables. Gorla [Gor08b] does not use the notion of uniformity, but in his first setting the separation result between π_{mix} and π_{sep} does also assume homomorphical translation of the parallel operator. Moreover, he specifies name invariance as a criterion for a good encoding, which is a more complex condition than Palamidessi's second condition. It turns out that, in our setting, we do not need a second condition like renaming preservation or name invariance, because we base our counterexamples in the following separation results on symmetric networks of the form $P \mid P$ as already Gorla did in [Gor08b]. For us, an encoding is uniform iff it translates the parallel operator homomorphically.

Definition 5.1 (Uniform encoding). An encoding $[\![\cdot]\!]$ from π_{mix} into an other language is a *uniform encoding* if and only if for all $P, Q \in \mathscr{P}_{\text{mix}}$

$$[\![\, P \mid Q \,]\!] = [\![\, P \,]\!] \mid [\![\, Q \,]\!] \tag{U}$$

(End of Definition 5.1)

[3] Remember that π_{sep} is a subset of π_{mix} and with it π_{mix} is at least as expressive as π_{sep}.

Actually, Theorem 4.4 should suffice to prove that there can not be a uniform and reasonable encoding from π_{mix} into π_{sep}, because uniform encodings preserve symmetries and it is possible to break symmetries in π_{mix} while this is not possible in π_{sep}. The crux is that there is no commonly accepted notion of reasonableness. For separation results, we seek a definition of reasonableness that is as weak as possible. But, without any notion of reasonableness, the theorem would not hold, because there are uniform encodings from π_{mix} into π_{sep}. For instance, we could simply translate everything to $\mathbf{0}$. Of course such an encoding makes no sense and so hardly anyone would call it reasonable. Usually, an encoding is called reasonable if it preserves some kind of behaviour or the ability to solve some kind of problem so to ensure that the purpose of the original term is preserved. In the following, we consider three different notions of reasonableness.

Version 1 For Palamidessi, an encoding is reasonable if it preserves the relevant observables and termination properties [Pal03]. Implicitly, she requires that a reasonable encoding should at least preserve the ability to solve leader election. We do alike but with a different interpretation of what it means to solve leader election that is more closely related to the definition used by Bougé [Bou88]: A network is said to solve leader election iff in each execution exactly one process propagates itself as leader while all the other processes propagate themselves as slaves. We assume the existence of two different predetermined output actions, one to propagate as leader and the other to propagate as slave. Moreover, we require that for both output actions neither the channel names nor the sent values are bound within the network[4]. The main difference to the definition of leader election used in [Pal03] is that here the slaves do not have to know the identity, i.e., the index, of the leader. So, this definition is usually considered as a weaker notion of the leader election problem. An encoding is now said to be reasonable iff it preserves the ability to solve the leader election problem.

Definition 5.2 (1-Reasonableness). An encoding $[\![\cdot]\!] : \mathscr{P}_{\text{mix}} \to \mathscr{P}_{\text{sep}}$ is *1-reasonable*, if $[\![P]\!]$ solves leader election if and only if P solves leader election for all $P \in \mathscr{P}_{\text{mix}}$. (End of Definition 5.2)

To prove that there is no uniform and reasonable encoding we force our encoding to lead to a network of two processes that is symmetric with respect to identity. By Theorem 4.4, this network has at least one symmetric execution. Because we use the identity as symmetry relation, in the symmetric execution both processes behave exactly the same such that if one of them propagates himself as leader then the other one does alike, which contradicts leader election.

Theorem 5.3 (Separation Result). *There is no uniform and 1-reasonable encoding from π_{mix} into π_{sep}.*

Proof of Theorem 5.3. Let us assume the contrary, i.e., there is a uniform and 1-reasonable encoding $[\![\cdot]\!]$ from π_{mix} into π_{sep}. Consider the network:

$$N \triangleq P \mid P \quad \text{with} \quad P \triangleq a.\overline{slave} + \overline{a}.\overline{leader}$$

Obviously $\sigma = \mathbf{id}$ is a symmetry relation of degree 2 and so $N = \left[a.\overline{slave} + \overline{a}.\overline{leader}\right]^2_\sigma$ is a symmetric network. Moreover N solves leader election, because the leader sends an empty message over channel *leader* and all slaves send an empty message over channel *slave*. By Definition 5.1 of uniformity, we have $[\![P \mid P]\!] \overset{(\mathrm{U})}{=} [\![P]\!] \mid [\![P]\!] = \left[[\![P]\!] \right]^2_{\mathbf{id}}$, i.e., $[\![N]\!]$ is again a symmetric network of degree 2 with **id** as symmetry relation. By Theorem 4.4, $[\![N]\!]$ has at least one symmetric execution and by reasonableness $[\![N]\!]$ must solve leader election, i.e., there is exactly one process that propagates itself as leader by an

[4] Note that if we allow bound names in these output actions, we could hardly predetermine them.

output action. Let μ_l denote this send action. By Definition 4.2, a symmetric execution has symmetric sequences of actions, i.e., the action μ_l is coupled to its symmetric counterpart building the sequence $[\mu_l]^{2,\tilde{z}'}_{\sigma'}$ for some $\tilde{z}' \in \mathscr{T}(\mathscr{N})$ and $\sigma' \in \mathrm{Sym}(2,\mathscr{N})$. By construction in the proof of Lemma 4.5, and because we start with **id**, we know that σ' consists of (permutations of) names that are bound in $[\![N]\!]$ or fresh. Because, by definition, μ_l can neither contain fresh nor bound names, we conclude $[\mu_l]^{2,\tilde{z}'}_{\sigma'} = \mu_l, \mu_l$, i.e., the output action appears twice in the symmetric execution. With that two processes propagate themselves as leader, which is a contradiction. □

Note that, in contrast to the proof of Palamidessi [Pal03, VPP07], we do not have to assume that the encoding is renaming preserving.

Version 2 Here, we first introduce a technical lemma. Intuitively, it states that the symmetric execution of a symmetric network of degree *n*, where *n* is *not* the minimal degree of the corresponding symmetry relation, can be subdivided into symmetric executions on symmetric subnetworks of the original network.

Lemma 5.4. *Let $\left[P_0\right]^{n,\tilde{x}}_{\sigma}$ be a symmetric network in $\mathscr{P}_{\mathrm{sep}}$. If the degree of σ is not minimal, i.e., if there is a $n' \in \mathbb{N}$ with $0 < n' < n$ such that $\sigma^{n'} = \mathbf{id}$, then $\left[P_0\right]^{n,\tilde{x}}_{\sigma}$ has a finite or an infinite symmetric execution*

$$\left[P_0\right]^{n,\tilde{x}}_{\sigma} \xrightarrow{[\mu_1]^{n,\tilde{x}}_{\sigma_1}} \left[P_1\right]^{n,\tilde{x}_1}_{\sigma_1} \xrightarrow{[\mu_2]^{n,\tilde{x}_1}_{\sigma_2}} \ldots \xrightarrow{[\mu_m]^{n,\tilde{x}_{m-1}}_{\sigma_m}} \left[P_m\right]^{n,\tilde{x}_m}_{\sigma_m} \not\to \quad \text{or} \quad \left[P_0\right]^{n,\tilde{x}}_{\sigma} \xrightarrow{[\mu_1]^{n,\tilde{x}}_{\sigma_1}} \left[P_1\right]^{n,\tilde{x}_1}_{\sigma_1} \xrightarrow{[\mu_2]^{n,\tilde{x}_1}_{\sigma_2}} \ldots$$

for a $m \in \mathbb{N}$, $P_1, \ldots, P_m \in \mathscr{P}_{\mathrm{sep}}$, $\sigma_1, \ldots, \sigma_m \in \mathrm{Sym}(n,\mathscr{N})$ with $\sigma \subseteq \sigma_1 \subseteq \ldots \subseteq \sigma_m$, $\tilde{x}_1, \ldots, \tilde{x}_m \in \mathscr{T}(\mathscr{N})$ and $\mu_1, \ldots, \mu_m \in \mathscr{L}$ or some $P_1, P_2, \ldots \in \mathscr{P}_{\mathrm{sep}}$, $\sigma_1, \sigma_2, \ldots \in \mathrm{Sym}(n,\mathscr{N})$ with $\sigma \subseteq \sigma_1 \subseteq \sigma_2 \subseteq \ldots$, some $\tilde{x}_1, \tilde{x}_2, \ldots \in \mathscr{T}(\mathscr{N})$ and $\mu_1, \mu_2, \ldots \in \mathscr{L}$ respectively such that $\left[P_0\right]^{n',\tilde{x}}_{\sigma}$ has the finite or infinite symmetric execution

$$\left[P_0\right]^{n',\tilde{x}'}_{\sigma} \xrightarrow{[\mu'_1]^{n',\tilde{x}'}_{\sigma'_1}} \left[P_1\right]^{n',\tilde{x}'_1}_{\sigma'_1} \xrightarrow{[\mu'_2]^{n',\tilde{x}'_1}_{\sigma'_2}} \ldots \xrightarrow{[\mu'_m]^{n',\tilde{x}'_{m-1}}_{\sigma'_m}} \left[P_m\right]^{n',\tilde{x}'_m}_{\sigma'_m} \not\to \quad \text{or} \quad \left[P_0\right]^{n',\tilde{x}}_{\sigma} \xrightarrow{[\mu'_1]^{n',\tilde{x}}_{\sigma'_1}} \left[P_1\right]^{n',\tilde{x}'_1}_{\sigma'_1} \xrightarrow{[\mu'_2]^{n',\tilde{x}'_1}_{\sigma'_2}} \ldots$$

for some $\tilde{x}'_1, \ldots, \tilde{x}'_m \in \mathscr{T}(\mathscr{N})$, $\mu'_1, \ldots, \mu'_m \in \mathscr{L}$ and $\sigma'_1, \ldots, \sigma'_m \in \mathrm{Sym}(n',\mathscr{N})$ with $\sigma \subseteq \sigma'_1 \subseteq \ldots \subseteq \sigma'_m$ or some $\tilde{x}'_1, \tilde{x}'_2, \ldots \in \mathscr{T}(\mathscr{N})$, $\mu'_1, \mu'_2, \ldots \in \mathscr{L}$ and $\sigma'_1, \sigma'_2, \ldots \in \mathrm{Sym}(n',\mathscr{N})$ with $\sigma \subseteq \sigma'_1 \subseteq \sigma'_2 \subseteq \ldots$ respectively such that $\tilde{x}'$ is a subsequence of $\tilde{x}$, $\tilde{x}'_i$ is a subsequence of $\tilde{x}_i$ and either μ'_i or if μ'_i is a bound output its unbound variant is in $[\mu_i]^{n,\tilde{x}_{i-1}}_{\sigma_i}$ for all $i \in \{1, \ldots, m\}$ or $i \in \mathbb{N}$ respectively.

Note that, like Theorem 4.4, this result is absolute in the sense that it holds independently of any notion of uniformity or reasonableness. We only present a proof sketch here. A proof can be found in [PN10].

Proof Sketch of Lemma 5.4. Assume there is a $0 < n' < n$ such that $\sigma^{n'} = \mathbf{id}$. Then because $\sigma^n = \mathbf{id}$ there must be a $k \in \mathbb{N}$ such that $n = k * n'$. Because $\sigma^0 = \sigma^{n'} = \sigma^{i*n'}$ for each $i \in \{1, \ldots, k\}$ we have $\sigma^j = \sigma^{j+n'}$. So $\left[P_0\right]^{n,\tilde{x}}_{\sigma}$ can be divided into k identical symmetric networks such that $\left[P_0\right]^{n,\tilde{x}}_{\sigma} = \left[P_0\right]^{n',\tilde{x}}_{\sigma} \mid \ldots \mid \left[P_0\right]^{n',\tilde{x}}_{\sigma}$ and $[\mu_1]^{n,\tilde{x}}_{\sigma_1}$ can be divided into k identical sequences $[\mu'_1]^{n',\tilde{x}'}_{\sigma'_1}$ for each $P_0 \in \mathscr{P}_{\mathrm{sep}}$ and each $\mu_1 \in \mathscr{L}$.

The proof is by induction on the number of sequences of n steps from a symmetric network to a symmetric network. To prove the inductive step, we perform a case analysis on whether the first step of such a sequence is due to an action of only one process of the network or to a communication between two processes. □

Gorla [Gor08b] defines the reasonableness of an encoding by the properties operational correspondence, divergence reflection and success sensitiveness. We use just the last of his properties instantiated with must testing. So we implicitly require divergence reflection. According to [Gor08b], success is represented by a process $\surd$ that is part of the source and the target language of the encoding and always appears unbound. More precisely, a process must-succeeds if it *always* reduces to a process containing a top-level unguarded occurrence of $\surd$. The fact that P must-succeeds is denoted by $P \downarrow\downarrow$. With it, an encoding is reasonable if the encoding of a term must-succeeds iff the term itself must-succeeds.

Definition 5.5 (2-Reasonableness). An encoding $[\![\cdot]\!] : \mathscr{P}_{\text{mix}} \to \mathscr{P}_{\text{sep}}$ is *2-reasonable*, if $P \downarrow\downarrow$ iff $[\![P]\!] \downarrow\downarrow$ for all $P \in \mathscr{P}_{\text{mix}}$. (End of Definition 5.5)

Again, we choose a term such that the encoding results in a network of the form $Q \mid Q$ in $\mathscr{P}_{\text{sep}}$ that is symmetric with respect to identity. In this case, we take advantage of the fact that the minimal degree of **id** is less than the degree of the network such that we can use Lemma 5.4 to subdivide the symmetric execution. With it already Q can perform the same sequence of steps as each process in $Q \mid Q$ performs in the symmetric execution.

Theorem 5.6 (Separation Result). *There is no uniform and 2-reasonable encoding from π_{mix} into π_{sep}.*

Proof of Theorem 5.6. Let us assume the contrary, i.e., there is a uniform and 2-reasonable encoding $[\![\cdot]\!]$ from π_{mix} into π_{sep}. Consider the network:

$$N \triangleq P \mid P \quad \text{with} \quad P \triangleq a.\mathbf{0} + \overline{a}.\surd$$

Obviously, $\sigma = \mathbf{id}$ is a symmetry relation of degree 2 and so $N = \left[a.\mathbf{0} + \overline{a}.\surd\right]^2_\sigma$ is a symmetric network. Moreover, we have $N \downarrow\downarrow$ but $P \not\downarrow$. We have $[\![P \mid P]\!] \overset{(\mathrm{U})}{=} [\![P]\!] \mid [\![P]\!] = \left[[\![P]\!] \right]^2_{\mathbf{id}}$, i.e., $[\![N]\!]$ is again a symmetric network of degree 2 with **id** as symmetry relation. By Theorem 4.4, $[\![N]\!]$ has at least one symmetric execution and by success sensitiveness and must testing $[\![N]\!]$ must reduce to a process containing a top-level unguarded occurrence of $\surd$ within this symmetric execution, i.e., there is a sequence of actions $\tilde{\mu} \in \mathscr{T}(\mathscr{L})$, a process $P' \in \mathscr{P}_{\text{sep}}$, a $\sigma' \in \mathrm{Sym}(2, \mathscr{N})$ and a sequence of names $\tilde{x}$ such that $[\![P]\!] \mid [\![P]\!] \xrightarrow{\tilde{\mu}} \left[P'\right]^{2,\tilde{x}}_{\sigma'}$ and P' or $\sigma'(P')$ contain a top-level unguarded occurrence of $\surd$. Then, by symmetry, both processes of $\left[P'\right]^{2,\tilde{x}}_{\sigma'}$ contain a top-level unguarded occurrence of $\surd$. By Lemma 5.4, there is a sequence of actions $\tilde{\mu}' \in \mathscr{T}(\mathscr{L})$ and an execution $[\![P]\!] \xrightarrow{\tilde{\mu}'} (\nu \tilde{x}') P'$ for a subsequence $\tilde{x}'$ of $\tilde{x}$. With it, $[\![P]\!] \downarrow\downarrow$, and with success sensitiveness $P \downarrow\downarrow$, which is a contradiction. □

Note that, reconsidering the proofs of this separation result in [Gor08b], we managed to omit one of Gorla's additional assumptions[5]. Moreover, note that because we focus on breaking symmetries instead of leader election, we can apply Theorem 4.4 to problem instances different from leader election.

Version 3 In his proofs of this separation result in [Gor08b] Gorla uses may testing to show that there are terms $P \in \mathscr{P}_{\text{mix}}$ such that $P \not\to$, $P \not\downarrow$ and $(P \mid P) \downarrow\downarrow$, but there are no such terms in $\mathscr{P}_{\text{sep}}$. Implicitly, he uses the fact that $P \not\downarrow$ and $(P \mid P) \downarrow\downarrow$ implies $P \mid P \to$ and that there are no terms P in $\mathscr{P}_{\text{sep}}$ such that $P \not\to$ and $P \mid P \to$. By proving this fact directly, we do not need any notion of testing to prove the separation result.

[5] Namely, we do not need the assumption that $\asymp_2$ is exact (first setting in [Gor08b]) or reduction sensitive (second setting in [Gor08b]) and we do not need to assume the stronger version of operational correspondence in the third setting in [Gor08b]. On the other side Gorla does not need to assume homomorphical translation of $\mid$ in his second and third setting. He uses the weaker notion of compositional translation of $\mid$ instead.

Definition 5.7 (3-Reasonableness). An encoding $[\![\cdot]\!] : \mathscr{P}_{\text{mix}} \to \mathscr{P}_{\text{sep}}$ is *3-reasonable* if $P \to$ if and only if $[\![P]\!] \to$ for all $P \in \mathscr{P}_{\text{mix}}$. (End of Definition 5.7)

As far as we know, only few intuitively reasonable encodings are not also 3-reasonable.

Again, for the separation proof, we enforce that the encoding results in a symmetric network $Q \mid Q$. By subdividing the symmetric execution of this network, we prove that $Q \to$ iff $Q \mid Q \to$, which does not necessarily hold in π_{mix}.

Theorem 5.8 (Separation Result). *There is no uniform and 3-reasonable encoding from π_{mix} into π_{sep}.*

Proof of Theorem 5.8. Let us assume the contrary, i.e., there is a uniform and 3-reasonable encoding $[\![\cdot]\!]$ from π_{mix} into π_{sep}. Consider the network:

$$N \triangleq P \mid P \quad \text{with} \quad P \triangleq a + \overline{a}$$

Obviously, $\sigma = \mathbf{id}$ is a symmetry relation of degree 2 and so $N = \left[a + \overline{a}\right]^2_\sigma$ is a symmetric network. Moreover, we have $N \to$ but $P \not\to$. We have $[\![P \mid P]\!] \overset{(\mathrm{U})}{=} [\![P]\!] \mid [\![P]\!] = \left[[\![P]\!] \right]^2_{\mathbf{id}}$, i.e., $[\![N]\!]$ is again a symmetric network of degree 2 with **id** as symmetry relation. By Theorem 4.4 $[\![N]\!]$ has at least one symmetric execution and by 3-reasonableness we have $[\![P]\!] \mid [\![P]\!] \to$ and $[\![P]\!] \not\to$. By Lemma 4.5, $[\![P]\!] \mid [\![P]\!] \to$ implies that there is at least one step in the symmetric execution, i.e., there is a $\mu \in \mathscr{L}$, a process $P' \in \pi_{\text{sep}}$, a $\sigma' \in \text{Sym}(2, \mathscr{N})$ and a sequence of names $\tilde{x} \in \mathscr{T}(\mathscr{N})$ such that $[\![P]\!] \mid [\![P]\!] \xrightarrow{[\mu]^2_{\sigma'}} \left[P'\right]^{2,\tilde{x}}_{\sigma'}$. By Lemma 5.4, there is a execution $[\![P]\!] \xrightarrow{\mu'} \left(\nu \tilde{x'}\right) P'$ for a subsequence $\tilde{x'}$ of $\tilde{x}$, $\mu' \in [\mu]^2_{\sigma'}$, which is a contradiction. □

Note that in opposite to both Palamidessi and Gorla we do not even assume divergence reflection.

6 Conclusion and Future Work

We prove that π_{mix} is strictly more expressive than π_{sep} by means of an absolute separation result about the ability to break initial symmetries. This result is independent of any notion of encodability, uniformity and reasonableness. By choosing the problem of breaking initial symmetries instead of leader election, we may significantly weaken the underlying definition of symmetry in comparison to [Pal03]. Moreover, we could still apply our absolute separation result to derive that there is no uniform and reasonable encoding from π_{mix} into π_{sep} considering three different definitions of reasonableness. It turns out that the concentration on the underlying problem of breaking initial symmetries allows us to use counterexamples different from leader election to prove the translational separation results. Likewise, the separation result in the setting of [Gor08b] can be derived by our absolute separation result as well. Besides that, our absolute separation result allows us to weaken the definition of uniformity in comparison to the translational separation result of [Pal03], and also to weaken the definition of reasonableness in comparison to the translational separation result in the first setting of [Gor08b]. Moreover, considering our last translational separation result, we can even withdraw the assumption of divergence reflection.

Our own translational separation results, i.e., the proofs of the non-existence of a uniform and reasonable encoding for different definitions of reasonableness, follow similar lines of argument. The proofs argue by contradiction. First, a symmetric network of the form $P \mid P$ in $\mathscr{P}_{\text{mix}}$ with special features is presented. Second, we use the fact that uniformity, i.e., the homomorphic translation of the parallel operator, preserves essentials parts of the symmetric nature of $P \mid P$. Third, we apply Theorem 4.4 to conclude

with the existence of a symmetric execution. In two proofs, we then apply Lemma 5.4 to subdivide this symmetric execution. At last, we derive a contradiction between the additional information provided by the symmetric execution (and its subdivision) and the respective definition of reasonableness.

Note that we prove the absolute result without any precondition. We use different definitions of reasonableness for the translational results. The only constant precondition of the translational separation results is the definition of uniformity, i.e., the homomorphic translation of the parallel operator. This condition is crucial. Without it, we could not apply our absolute separation result. To the best of our knowledge, only Gorla ever managed to prove such a separation result between π_{mix} and π_{sep} without the homomorphic translation of the parallel operator, using compositionality, operational correspondence, divergence reflection, success sensitiveness and either a reduction sensitive version of $\asymp$ or the stronger version of operational correspondence of his third setting. However, Gorla believes that the result also holds for the general formulation of his criteria, i.e., without assuming a reduction sensitive version of $\asymp$ or the stronger version of operational correspondence of his third setting. We believe that this is an interesting open question.

We may also turn the non-existence of a uniform *and* reasonable encoding around and rephrase it as a weakened existence statement. Recall that any uniform encoding from π_{mix} into π_{sep} preserves symmetries. While it is possible to break such symmetries in π_{mix}, this is not possible in π_{sep}. Thus, should there be a non-uniform (at least: "weakly compositional") *but* reasonable encoding from π_{mix} into π_{sep}, then *it would have to be the encoding itself to break these symmetries*. Finding such a reasonable encoding is an open problem, if reasonableness includes divergence reflection. A uniform and "almost reasonable" divergent encoding was already presented in [Nes00].

References

[BGZ00] Nadia Busi, Roberto Gorrieri, and Gianluigi Zavattaro. On the expressiveness of linda coordination primitives. *Information and Compututation*, 156(1–2):90–121, 2000.

[Bou88] Luc Bougé. On the Existence of Symmetric Algorithms to Find Leaders in Networks of Communicating Sequential Processes. *Acta Informatica*, 25(4):179–201, Mai 1988.

[Gor08a] Daniele Gorla. Comparing Communication Primitives via their Relative Expressive Power. *Information and Computation*, 206(8):931–952, 2008.

[Gor08b] Daniele Gorla. Towards a Unified Approach to Encodability and Separation Results for Process Calculi. Technical report, Dip. di Informatica, Univ. di Roma "La Sapienza", 10 2008. To appear in Information and Computation.

[Nes00] Uwe Nestmann. What is a "Good" Encoding of Guarded Choice? *Information and Computation*, 156(1-2):287–319, 2000.

[Pal03] Catuscia Palamidessi. Comparing the Expressive Power of the Synchronous and the Asynchronous π-calculi. *Mathematical Structures in Computer Science*, 13(5):685–719, 2003.

[Par08] Joachim Parrow. Expressiveness of Process Algebras. *Electronic Notes in Theoretical Computer Science*, 209:173–186, 2008.

[PN10] Kirstin Peters and Uwe Nestmann. Breaking symmetries. Technical report, Technische Universität Berlin, Germany, July 2010. http://arxiv.org/corr/home.

[SW01] Davide Sangiorgi and David Walker. *The π-calculus: A Theory of Mobile Processes*. Cambridge University Press New York, NY, USA, October 16 2001.

[VPP07] Maria Grazia Vigliotti, Iain Phillips, and Catuscia Palamidessi. Tutorial on separation results in process calculi via leader election problems. *Theoretical Computer Science*, 388(1–3):267–289, December 5 2007.