Comparing Process Calculi Using Encodings

Kirstin Peters

TU Berlin/TU Darmstadt

Encodings or the proof of their absence are the main way to compare process calculi. To analyse the quality of encodings and to rule out trivial or meaningless encodings, they are augmented with encodability criteria. There exists a bunch of different criteria and different variants of criteria in order to reason in different settings. This leads to incomparable results. Moreover, it is not always clear whether the criteria used to obtain a result in a particular setting do indeed fit to this setting. This paper provides a short survey on often used encodability criteria, general frameworks that try to provide a unified notion of the quality of an encoding, and methods to analyse and compare encodability criteria.

1 Introduction

Process calculi (or process algebra) is one area of formalisations of concurrent systems. Other areas are for example Petri nets [54] or the Actor model [25]. *A brief history of process algebra* can be found in [2]. Over the time different process calculi emerge. Examples are CCS [32], the π-calculus [36, 35], CSP [26], or ACP [4], just to name some of the most prominent ones. Note that each of these calculi denotes rather a family of process calculi. The number of different process calculi is in fact enormous. As discussed in [41] there are many good reasons for this great number of different calculi. The most plausible is maybe that many of these different calculi stem from different practical needs. They are designed as domain-specific calculi capturing exactly the set of primitives necessary to model the desired system at a proper level of abstraction without overloading the theory with (for this domain) unnecessary operations. The large number of calculi calls for methods to compare different calculi or different variants within a family of process calculi with respect to their expressive power. The most prominent such method is language embedding using encodings [5]. Encodings—or the proof of their absence—do not only allow to compare the expressive power of languages but also formalise similarities and differences between the considered languages. So they provide a base for implementations of languages into real systems.

There are basically two ways to measure the expressive power of a language [45].

An *absolute result* is a result about the expressive power of a single process calculus, usually obtained by proving the ability (*positive absolute result*) or inability (*negative absolute result*) to solve some kind of problem (see [45, 22] and even [31]), whereas a *relative result* compares two process calculi.

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. Note that combining two absolute results that are both positive or both negative usually does not reveal much information, because it proves only that the considered languages do not differ with respect to the respective particular problem. 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. The terminology of the relative expressive power has also been attributed (see [45, 22]) 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. Indeed, encodings are the main way to relate the expressive power of two process calculi.

A positive relative result, i.e., the proof of the existence of an encoding, is denoted as *encodability result*, whereas a negative result is denoted as *separation result*.

Language comparison by means of encodings is a wide area of research within the context of process calculi. Reasonable and meaningful encodings from one language into another show that the latter is at least as expressive as the former, whereas the absence of such an encoding shows that the former can express some behaviour that is not expressible in the latter, i.e., reveals a difference in the expressive power of the former compared to the latter language. To analyse the quality of encodings and to rule out trivial or meaningless encodings, they are augmented with encodability criteria. However, as stated several times in literature (e.g. in [43, 41, 45, 22]), there is no agreement on what set of criteria makes an encoding reasonable and meaningful. Sometimes it is even stated that such an agreement may not exist or may not be desirable (see e.g. [43]), because many criteria result from different practical needs. They are often derived from the main purpose of the current analysis. From a practical point of view this is meaningful. But, obviously, using different quality criteria for different results, because they were motivated by different practical problems, naturally leads to incomparable results.

After some technical preliminaries in Section 2, Section 3 provides a short survey of often used encodability criteria. This section is the heart of this paper. Then, Section 4 shortly outlines three approaches that try to overcome the problem of incomparable results and provide a unified notion of the quality of an encoding. Similarly, Section 5 shortly outlines the only way to formally analyse and compare encodability criteria we are aware of. Finally, Section 6 raises some open questions. Note that this paper provides a rather abstract view on encodings and focuses on encodability criteria. A more practical study of encodings, including a discussion of how to obtain an encoding and prove it correct as well as some actual examples of encodings can be found in [49].

2   Process Calculi and Encodings

A *process calculus* is a language $\mathscr{L}_C = (\mathscr{P}_C, \longmapsto_C)$ consisting of a set of terms $\mathscr{P}_C$—its *syntax*—and a relation on terms $\longmapsto_C \subseteq \mathscr{P}_C \times \mathscr{P}_C$—its *semantics*. The elements of $\mathscr{P}_C$ are called *process terms* or shortly processes or terms. We use upper case letters $P, Q, S, T, \ldots$ to range over process terms.

The *syntax* is usually defined by a context-free grammar defining operators. An *operator* is a function from sorts and process terms into a process term. Sorts can be used to introduce data values or structural information such as location names or identities. For simplicity, we restrict our attention to a single such sort. Assume a countably-infinite set $\mathscr{N}$, whose elements are called *names*. Names are the universe of elements of which the processes are constructed within many process calculi such as e.g. in the π-calculus. Then an operator is a function $\mathrm{op} : \mathscr{N}^n \times \mathscr{P}_C^m \to \mathscr{P}_C$ from names and process terms into a process term. In this case, we say op has the *arity m*. Sometimes, process calculi also specify operators that, instead of a fixed number of arguments, accept any finite set of names and/or process terms usually indexed by a finite index set I. In this case, the arity of the operator is not a fixed value but, for a given set of arguments, is determined by the number of process terms among the arguments. An example of such an operator is given by the choice operator in the π-calculus. Operators of arity 0 are denoted as *constants*.

The semantics of a process calculus can be given as a reduction semantics—that describes the interaction within a system of processes—or a labelled semantics—that also describes the interactions of such a system with its environment. Here we assume that the semantics of the language is provided as a reduction semantics, because in the context of encodings the treatment of reductions is simpler. A *step* $P \longmapsto_C P'$ is an element $(P, P') \in \longmapsto_C$. Let $P \longmapsto_C$ denote the existence of a step from P, i.e.,

$\exists P' \in \mathscr{P}_C. P \longmapsto_C P'$, let $P \longmapsto\!\!\!\!\!/_C$ denote the absence of a step from P, i.e., $\neg (P \longmapsto_C)$, and let $\Longmapsto_C$ denote the reflexive and transitive closure of $\longmapsto_C$. We write $P \longmapsto_C{}^\omega$ if P can do an infinite sequence of steps. A term P such that $P \longmapsto_C{}^\omega$ is called *divergent*.

Intuitively, a *context* $\mathscr{C}([\cdot]_1, \ldots, [\cdot]_n) : \mathscr{P}_C^n \to \mathscr{P}_C$ is a process term with n holes. Formally, it is a function from n process terms into a process term, i.e., given n terms $P_1, \ldots, P_n \in \mathscr{P}_C$, the term $\mathscr{C}(P_1, \ldots, P_n)$ is the result of inserting the n terms $P_1, \ldots, P_n$ in that order into the n holes of $\mathscr{C}$. We also consider contexts $\mathscr{C}([\cdot]_1, \ldots, [\cdot]_n, [\cdot]_{n+1}, \ldots, [\cdot]_{n+m}) : \mathscr{N}^n \times \mathscr{P}_C^m \to \mathscr{P}_C$ that have holes for names and terms.

A typical operator is the restriction of scopes of names. A *scope* defines an area in which a particular name is known and can be used. For several reasons, it can be useful to restrict the scope of a name. For instance to forbid interactions between two processes or with an unknown and, hence, potentially untrusted environment. Names whose scope is restricted such that they cannot be used from outside the scope are denoted as *bound names*. The remaining names are called *free names*. Accordingly, we assume three sets—the sets of names $n(P)$ and its subsets of free names $fn(P)$ and bound names $bn(P)$— for each term P. In the case of bound names, their syntactical representation as lower case letter serves as a place holder for any fresh name, i.e., any name that does not occur elsewhere in the term. To avoid name capture or clashes, i.e., to avoid confusion between free and bound names or different bound names, bound names can be mapped to fresh names by α-*conversion*. We write $P \equiv_\alpha Q$ if P and Q differ only by α-conversion. A *substitution* σ 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. The application of a substitution to a term $P\{ y_1/x_1, \ldots, y_n/x_n \}$ is defined as the result of simultaneously replacing all free occurrences of x_i by y_i for $i \in \{ 1, \ldots, n \}$, possibly applying α-conversion to avoid capture or name clashes. For all names $\mathscr{N} \setminus \{ x_1, \ldots, x_n \}$ the substitution behaves as the identity mapping.

To compare process terms, process calculi usually come with different well-studied preorders and equivalences (see [16, 14] for an overview and a classification of the most frequent equivalences). A special kind of equivalence with great importance to reason about processes are congruences. A congruence is the closure of an equivalence with respect to contexts, i.e., an equivalence $\mathscr{R} \subseteq \mathscr{P}_C \times \mathscr{P}_C$ is a *congruence* if $(P, Q) \in \mathscr{R}$ implies $(\mathscr{C}(P), \mathscr{C}(Q)) \in \mathscr{R}$ for all terms $P, Q \in \mathscr{P}_C$ and all contexts $\mathscr{C}([\cdot]) : \mathscr{P}_C \to \mathscr{P}_C$. Moreover, let C be a set of $\mathscr{P}_C \to \mathscr{P}_C$-contexts, then $\mathscr{R} \subseteq \mathscr{P}_C \times \mathscr{P}_C$ is a *congruence with respect to* C if $(P, Q) \in \mathscr{R}$ implies $(\mathscr{C}(P), \mathscr{C}(Q)) \in \mathscr{R}$ for all terms $P, Q \in \mathscr{P}_C$ and all contexts $\mathscr{C} \in C$. Moreover, process calculi usually come with a special congruence $\equiv_C \subseteq \mathscr{P}_C \times \mathscr{P}_C$ called *structural congruence*. Its main purpose is to equate syntactically different process terms that model quasi-identical behaviour.

Of special interest are simulation relations; in particular bisimulations [57]. $\mathscr{R}$ is a bisimulation if any two related processes mutually simulate their respective sequences of steps, such that the derivatives are again related.

Definition 2.1 (Bisimulation) $\mathscr{R}$ *is a* (weak reduction) bisimulation *if for each* $(P, Q) \in \mathscr{R}$:
- $P \Longmapsto_C P'$ *implies* $\exists Q'. Q \Longmapsto_C Q' \wedge (P', Q') \in \mathscr{R}$
- $Q \Longmapsto_C Q'$ *implies* $\exists P'. P \Longmapsto_C P' \wedge (P', Q') \in \mathscr{R}$

Two terms are bisimilar *if there exists a bisimulation that relates them.*

The definition of a *strong (reduction) bisimulation* is obtained by replacing all $\Longmapsto_C$ by $\longmapsto_C$ in the above definition, i.e., a strong bisimulation requires that a step has to be simulated by a single step. Coupled similarity (defined below) is strictly weaker than bisimilarity. As pointed out in [47], in contrast to bisimilarity it allows for intermediate states in simulations: states that cannot be identified with states of the simulated term. Each symmetric coupled simulation is a bisimulation.

Definition 2.2 (Coupled Simulation) *A relation $\mathscr{R}$ is a* (weak reduction) coupled simulation *if, for each* $(P,Q) \in \mathscr{R}$ *with* $P \Longmapsto_C P'$, $(\exists Q'. Q \Longmapsto_C Q' \wedge (P',Q') \in \mathscr{R})$ *and* $(\exists Q'. Q \Longmapsto_C Q' \wedge (Q',P') \in \mathscr{R})$. *Two terms are* coupled similar *if they are related by a coupled simulation in both directions.*

An *encoding* from $\mathscr{L}_S = (\mathscr{P}_S, \longmapsto_S)$ into $\mathscr{L}_T = (\mathscr{P}_T, \longmapsto_T)$ relates two process calculi. We call $\mathscr{L}_S$ the *source* and $\mathscr{L}_T$ the *target language*. Accordingly, terms of $\mathscr{P}_S$ are *source terms* and of $\mathscr{P}_T$ *target terms*. In the simplest case an encoding from $\mathscr{L}_S$ into $\mathscr{L}_T$ is an *encoding function* $[\![\cdot]\!] : \mathscr{P}_S \to \mathscr{P}_T$ from source terms into target terms. Sometimes an encoding is defined by several functions, such as the encoding function and the renaming policy (see below). Else we identify an encoding with its encoding function.

Sometimes it is necessary to translate a source term name into a sequence of names or reserve some names for the encoding function. To ensure that there are no conflicts between these reserved names and the source term names, [22] equips an encoding with a *renaming policy* ϕ, i.e., a substitution from names into sequences of pairwise disjoint names. To keep distinct names distinct, [22] assumes that the sequences of names that result from applying a renaming policy to distinct names have no common name. Moreover, if the renaming policy translates a single name into a sequence of names then the length of such a sequence has to be the same for all names, such that the encoding can not distinguish between different source term names by the length of the sequences to which they are encoded. Obviously, no name should be translated into an infinite sequence of names.

Definition 2.3 (Renaming Policy) *A substitution* $\phi : \mathcal{N} \to \mathcal{N}^n$ *from names into sequences of pairwise disjoint names is a renaming policy, if* $\forall x, y \in \mathcal{N} . x \neq y$ *implies* $\phi(x) \cap \phi(y) = \emptyset$, *where* $\phi(z)$ *is considered as a set.*

Note that the renaming policy allows us to use the names reserved by the encoding like implicit parameters. It is for instance possible that some part of the encoding introduces a free occurrence of a reserved name within the encoding of a subterm which is bound by the surrounding part of the encoding. Examples can be found in [22, 49]. Accordingly, in [22] an encoding is a pair $([\![\cdot]\!], \phi(\cdot))$.

An *encodability criterion* is a predicate on encoding functions, used to reason about the quality of encodings. To simplify the presentation we assume henceforth that $\mathscr{P}_S \cap \mathscr{P}_T = \emptyset$ and thus $\mathscr{P}_S \uplus \mathscr{P}_T = \mathscr{P}_S \cup \mathscr{P}_T$. We say that a condition $\mathsf{P} : (\mathscr{P}_S \uplus \mathscr{P}_T) \to \mathbb{B}$ is *preserved* by an encoding if for all source terms S that satisfy P, the condition P also holds for $[\![S]\!]$. A condition is *reflected* by an encoding if whenever $[\![S]\!]$ satisfies it, then so does S. Finally an encoding *respects* a condition if it both preserves and reflects it.

3 Encodability Criteria

Encodings are used to compare process calculi and to reason about their expressive power. Encodability criteria are conditions that limit the existence of encodings. Their main purpose is to rule out trivial or meaningless encodings, but they can also be used to limit attention to encodings that are of special interest in a particular domain or for a particular purpose. These quality criteria are the main tool in *separation results*, saying that one calculus is not expressible in another one; here one has to show that no encoding meeting these criteria exists. To obtain stronger separation results, care has to be taken in selecting quality criteria that are not too restrictive. For *encodability results*, saying that one calculus is expressible in another one, all one needs is an encoding, together with criteria testifying for the quality of the encoding. Here it is important that the criteria are not too weak.

In the literature various different criteria and different variants of the same criteria are employed to achieve separation and encodability results [15, 40, 42, 43, 44, 41, 59, 9, 45, 8, 22, 52, 17]. Some criteria, like full abstraction or operational correspondence, are used frequently. Other criteria are used to enforce a property of encodings that might only be necessary within a certain domain. For instance, the homomorphic translation of the parallel operator—in general a rather strict criterion—was used in [43] to show the absence of an encoding from the synchronous into the asynchronous π-calculus, because this requirement forbids for the introduction of global coordinators. Thus this criterion is useful when reasoning about the concurrent behaviour of processes, although it is in general too strict to reason about their interleaving behaviour. Unfortunately, it is not always obvious or clear whether the criteria used to obtain a result in a particular setting do indeed fit to this setting. Indeed, as discussed in [52], the homomorphic translation of the parallel operator forbids more than global coordinators, i.e., is too strict even in a concurrent setting (compare to Section 3.6).

In the following we shortly revise some of the most commonly used encodability criteria. In [39] also a class of rather quantitative criteria for the effectiveness or efficiency of an encoding is discussed. The efficiency of an encoding can be measured for example by a criterion to count the number of messages or steps of an emulation [7, 1, 29]. However, here we are more focused on semantic and structural criteria.

3.1 Direct Comparison via a Relation

The least debatable criterion is the direct comparison of the source and the target language by a behavioural equivalence (or preorder). This criterion requires that $\forall S \in \mathscr{P}_\mathrm{S}.\ [\![S]\!]\mathscr{R}S$ holds, for some behavioural equivalence (or preorder) $\mathscr{R} \subseteq (\mathscr{P}_\mathrm{S} \uplus \mathscr{P}_\mathrm{T}) \times (\mathscr{P}_\mathrm{S} \uplus \mathscr{P}_\mathrm{T})$ that is either defined in exactly the same way on the source and the target language (as e.g. in [45]) or explicitly distinguishes between source and target terms (as e.g. in [50]). Intuitively, it is required that the encoding respects the semantics of the source modulo the chosen relation. Obviously, the quality of the encoding directly depends on the choice of $\mathscr{R}$. A stricter such relation leads to stricter requirements on the encoding function. It is also very clear in this case under which circumstances two different results can be compared. If both results are proven with respect to the same relation then the results can be compared directly. If one of the considered relations is strictly weaker then the results can be compared with respect to the weaker relation. Else, if the relations are incomparable, also the results are incomparable. Hence, a language $\mathscr{L}_1$ can be considered as strictly weaker than the language $\mathscr{L}_2$ with respect to $\mathscr{R}$, if there is an encoding from $\mathscr{L}_1$ into $\mathscr{L}_2$ that satisfies the above requirement but there is no such encoding from $\mathscr{L}_2$ into $\mathscr{L}_1$.

Of course, there may be still some debate on how to choose the relation $\mathscr{R}$. For instance neither the identity nor $\mathscr{R} = (\mathscr{P}_\mathrm{S} \uplus \mathscr{P}_\mathrm{T}) \times (\mathscr{P}_\mathrm{S} \uplus \mathscr{P}_\mathrm{T})$ are intuitively meaningful choices. Moreover, as shown in [16, 14] there are usually very many potential candidates. Variants of bisimulation are usually a rather strong candidate suitable for encodability results, whereas weaker candidates such as trace equivalence are better suited for separation results. For a congruence it is sometimes necessary to restrict the set of considered contexts to contexts that respect the protocol behind the encoding (see e.g. [58, 18]). However, since the choice of the equivalence directly monitors the requirements on the encoding function, this problem is not that serious. There are, however, two serious drawbacks of this criterion. The first is the requirement that $\mathscr{R}$ is either defined in exactly the same way on the source and the target language or explicitly distinguishes between source and target terms. Usually behavioural relations are designed for one specific language, so it is not clear how to define a suitable relation if the two languages have no standard behavioural equivalence (or preorder) in common. A standard candidate for such an equivalence is weak reduction bisimulation, that is defined similarly on all calculi. Unfortunately, the variant of weak bisimulation given in Definition 2.1 is trivial. We need to add a condition that compares the observables

of bisimilar states in order to obtain a meaningful relation. But, since the source and target language might specify different standard observables, this equivalence (if it can be constructed at all) will usually be very complex and unreadable. Note that [22] solves this problem by using a special observable called success instead of the standard observables of the languages, whereas [12, 18] use relations to compare observables of different languages. As a consequence, if $\mathscr{R}$ is not a standard equivalence, the direct comparison of source and target terms modulo $\mathscr{R}$ reveals less intuition on the encoding function. The second problem is that a complex $\mathscr{R}$ leads to a hard proof of this criterion. If $\mathscr{R}$ is not a standard equivalence, most of the standard techniques that would ease such a proof may not be applicable.

3.2 Full Abstraction

Whenever source and target can not be compared directly with respect to a standard equivalence, full abstraction might be a way to nonetheless use standard equivalences. *Full abstraction*—denoted as *observational correspondence* in [13]—is probably the most common quality criterion for language comparison. It is used for instance in [55, 38, 56, 60, 42, 3, 48], just to name some. Full abstraction as proof method for language comparison was adapted from the use of full abstraction to show correspondence between a denotational semantics of a program and its operational semantics. An encoding $[\![\cdot]\!]$ is fully abstract if

$$\forall S_1, S_2 \in \mathscr{P}_{\mathrm{S}}.\ S_1 \mathscr{R}_{\mathrm{S}} S_2 \quad \text{iff} \quad [\![S_1]\!] \mathscr{R}_{\mathrm{T}} [\![S_2]\!]$$

for two behavioural equivalences $\mathscr{R}_{\mathrm{S}} \subseteq \mathscr{P}_{\mathrm{S}} \times \mathscr{P}_{\mathrm{S}}$ and $\mathscr{R}_{\mathrm{T}} \subseteq \mathscr{P}_{\mathrm{T}} \times \mathscr{P}_{\mathrm{T}}$, i.e., full abstraction requires that equivalent source terms have to be mapped into equivalent target terms and vice versa. Note that the direction from the left to the right is often called soundness condition and the only if part completeness condition of full abstraction. The soundness condition is usually the most demanding part. Note that some well-known and widely accepted encodings, as e.g. [6, 27, 33, 34], do not satisfy this property with respect to a reasonable combination of standard equivalences. The main advantage of full abstraction is its wide applicability also with respect to (more or less) standard equivalences. It does e.g. not require that source and target share any notion of observable, which is a premise for the use of most of the standard equivalences in the criterion above. However, again there may be a very large number of equivalences on the source as well as equivalences on the target and the strictness of the property expressed by full abstraction strongly relies on the combination of the chosen equivalences. To reduce the strong dependence of full abstraction results on the chosen equivalences, full abstraction is often combined with operational correspondence. In [13] it is even stated that full abstraction is not of much use without operational correspondence. The two papers [23, 46] come to a similar conclusion after discussing potential pitfalls and misunderstandings w.r.t. full abstraction. These papers also hint to earlier such discussions. The possibility to chose a combination of standard equivalence that turn full abstraction trivial is a major drawback of this criterion. Another major drawback is that, because of the various possibilities to choose these two equivalences, it is often not possible to compare different full abstraction results.

3.3 Operational Correspondence

Intuitively, *operational correspondence* requires executions to be respected. Again, it consists of a completeness and a soundness part. The completeness condition, also called adequacy, requires that for all source term steps $S \longmapsto_{\mathrm{S}} S'$ or source term executions $S \Longmapsto_{\mathrm{S}} S'$ there is one emulation in the target language such that $[\![S]\!] \Longmapsto_{\mathrm{T}} \asymp_{\mathrm{T}} [\![S']\!]$, where $\asymp_{\mathrm{T}} \subseteq \mathscr{P}_{\mathrm{T}} \times \mathscr{P}_{\mathrm{T}}$ is some equivalence on the target language. Note that there is no difference in the consideration of single source term steps or source term executions.

Intuitively, the completeness condition requires that any source term execution is emulated by the target term modulo some equivalence $\asymp_T$. Again, completeness is usually the easier part.

For the soundness condition we basically find two formulations. The stricter formulation requires that for all executions of the target $[\![S]\!] \Longmapsto_T T$ there exists some execution of the source $S \Longmapsto_S S'$ such that $[\![S']\!] \asymp_T T$. Intuitively, soundness requires that whatever $[\![S]\!]$ can do is a translation of some behaviour of S modulo $\asymp_T$. The weaker formulation requires that for all executions of the target $[\![S]\!] \Longmapsto_T T$ there exists some execution of the source $S \Longmapsto_S S'$ and some execution of the target $T \Longmapsto_T T'$ such that $[\![S']\!] \asymp_T T'$. Intuitively, it states that any execution of the target is some part of the emulation of an execution in the source modulo $\asymp_T$ [47, 22]. The main difference is that the latter formulation allows for intermediate or partial commitment states [47, 49, 24], i.e., for states that do not need to be related directly to the states of the respective source term but that have to belong to some emulation of a source term step. In this sense, an intermediate state results from the partial emulation of a source term step. In particular the last variant, proposed in [22], was used for numerous encodability and separation results.

Definition 3.1 (Operational Correspondence) *An encoding* $[\![\cdot]\!] : \mathscr{P}_S \to \mathscr{P}_T$ *is strongly operationally corresponding w.r.t.* $\asymp_T \subseteq \mathscr{P}_T^2$ *if it is:*
  Strongly Complete: $\forall S, S'. \, S \longmapsto_S S'$ *implies* $(\exists T. \, [\![S]\!] \longmapsto_T T \wedge ([\![S']\!], T) \in \asymp_T)$
  Strongly Sound: $\forall S, T. \, [\![S]\!] \longmapsto_T T$ *implies* $(\exists S'. \, S \longmapsto_S S' \wedge ([\![S']\!], T) \in \asymp_T)$
$[\![\cdot]\!] : \mathscr{P}_S \to \mathscr{P}_T$ *is operationally corresponding w.r.t.* $\asymp_T \subseteq \mathscr{P}_T^2$ *if it is:*
  Complete: $\forall S, S'. \, S \Longmapsto_S S'$ *implies* $(\exists T. \, [\![S]\!] \Longmapsto_T T \wedge ([\![S']\!], T) \in \asymp_T)$
  Sound: $\forall S, T. \, [\![S]\!] \Longmapsto_T T$ *implies* $(\exists S'. \, S \Longmapsto_S S' \wedge ([\![S']\!], T) \in \asymp_T)$
$[\![\cdot]\!] : \mathscr{P}_S \to \mathscr{P}_T$ *is weakly operationally corresponding w.r.t.* $\asymp_T \subseteq \mathscr{P}_T^2$ *if it is:*
  Complete: $\forall S, S'. \, S \Longmapsto_S S'$ *implies* $(\exists T. \, [\![S]\!] \Longmapsto_T T \wedge ([\![S']\!], T) \in \asymp_T)$
  Weakly Sound: $\forall S, T. \, [\![S]\!] \Longmapsto_T T$ *implies* $(\exists S', T'. \, S \Longmapsto_S S' \wedge T \Longmapsto_T T' \wedge ([\![S']\!], T') \in \asymp_T)$

Again different variants of operational correspondence may arise from different requirements on the assumed equivalence $\asymp_T$ on the target language and a wrong choice might turn operational correspondence trivial. In particular, each encoding is operational corresponding w.r.t. the universal relation on target terms.

In [13] a version of operational correspondence is presented, that considers labelled steps instead of a reduction semantics under the assumption that there exists a mapping $\widehat{\cdot}$ from the labels of the source term into the labels of the target term. Hence, the resulting requirement—$[\![S]\!] \overset{\widehat{\lambda}}{\Longrightarrow}_T \asymp_T [\![S']\!]$ whenever $S \overset{\lambda}{\Longrightarrow}_S S'$ and $[\![S]\!] \overset{\lambda}{\Longrightarrow}_T T$ implies $S \overset{\lambda'}{\Longrightarrow}_S S'$ for some λ', S' such that $[\![S']\!] \asymp_T T$ and $\widehat{\lambda'} = \lambda$—can be considered as stricter than the above variant of operational correspondence, because also observables have to be respected modulo $\widehat{\cdot}$. In fact, without this strengthening to labelled semantics, operational correspondence alone can hardly be considered as suitable criterion. Hence, the above version based on reduction semantics is usually combined with other criteria as full abstraction or some requirements on the preservation or reflection of some kind of observable.

3.4 Observables, Testing, and Termination

If source and target terms can not be compared directly by a standard equivalence, e.g. because not all standard observables of the source are standard observables of the target, a natural weaker requirement is to consider preservation or reflection of the remaining observables that are shared by source and target. Typical observables are links used for communication, barbs (communication capabilities), or traces [59, 45]. Moreover, the use of *termination properties* as the possibility of deadlock, livelock, or divergence

are popular [40, 45, 22, 13]. Also all kind of *tests* that the process may (or must) pass, for some formal notion of test can be used to compare source and target behaviour [45, 22].

Another kind of termination property is the promptness condition. Intuitively, promptness ensures that an encoding does not introduce preprocessing steps. An encoding $[\![\cdot]\!]$ is *prompt*, if $S \not\longmapsto_S$ implies $[\![S]\!] \not\longmapsto_T$ for all source terms $S \in \mathscr{P}_S$.

In [22] the criterion *success sensitiveness* was proposed. An encoding is success sensitive if it respects reachability of a particular process $\checkmark$ that represents successful termination, or some other form of success, and is added to the syntax of the source as well as the target language. Since $\checkmark$ cannot be further reduced and $\mathsf{n}(\checkmark) = \mathsf{fn}(\checkmark) = \mathsf{bn}(\checkmark) = \emptyset$, the semantics and structural congruence of a process calculus are not affected by this additional constant operator. We write $P \downarrow_\checkmark$ to denote the fact that P is successful—however this predicate might be defined in the particular source or target language. Reachability of success is then defined as $P \Downarrow_\checkmark \triangleq \exists P'. P \Longmapsto P' \wedge P' \downarrow_\checkmark$. We use may-testing here, but alternatively e.g. must-testing or fair-testing can also be implemented. An encoding is success sensitive if each source term and its translation answer the test for reachability of success in the same way.

Definition 3.2 (Success Sensitiveness) *Let $\mathscr{L}_S$ and $\mathscr{L}_T$ each define a predicate $\cdot\downarrow_\checkmark : \mathscr{P} \to \mathbb{B}$. An encoding $[\![\cdot]\!] : \mathscr{P}_S \to \mathscr{P}_T$ is* success sensitive *if, for all $S \in \mathscr{P}_S$, $S \Downarrow_\checkmark$ iff $[\![S]\!] \Downarrow_\checkmark$.*

Note that $\checkmark$ can be considered as some kind of termination property—describing successful termination in contrast to not successful termination by a term that cannot reduce but is not $\checkmark$—or as the successful pass of some kind of test.

If the source and target language share standard observables, we can easily extend success sensitiveness to barb sensitiveness. Assume that, instead of $\cdot\downarrow_\checkmark : \mathscr{P} \to \mathbb{B}$, the languages $\mathscr{L}_S$ and $\mathscr{L}_T$ each define a predicate $\cdot\downarrow\cdot : \mathscr{P} \times \mathscr{B} \to \mathbb{B}$. An encoding $[\![\cdot]\!] : \mathscr{P}_S \to \mathscr{P}_T$ *strongly respects barbs* if, for all $S \in \mathscr{P}_S$ and all $\alpha \in \mathscr{B}$, $S \downarrow \alpha$ iff $[\![S]\!] \downarrow \alpha$. A weak variant is obtained by replacing the predicate $\cdot\downarrow\cdot$ for the existence of a barb by a predicate $\cdot\Downarrow\cdot$ that is true if a barb is reachable.

3.5 Structural Requirements

The above discussed criteria describe semantic requirements, i.e., requirements on the behaviour of target terms. To prove the quality of an encoding semantic criteria are often combined with structural criteria. Intuitively, the semantic criteria describe how the encoded terms should behave with respect to the behaviour of the corresponding source term, whereas structural criteria rather describe how the encoded terms have to look like. Moreover, as stated in [45], structural criteria are needed in order to measure expressiveness of operators in contrast to expressiveness of terms.

The most common structural criterion is compositionality with homomorphy as a special case. Intuitively, *compositionality* states that the translation of a compound term must be defined in terms of the translation of the subterms. To mediate between translations of subterms, a context is introduced. Different manifestations result from different requirements on allowed contexts. In the strictest form, often denoted as *homomorphy*, the context has to be the original source term operator again, i.e., an encoding translates the source term operator $\mathsf{op}(x_1, \ldots, x_n, S_1, \ldots, S_m)$ homomorphically if it ensures that $[\![\mathsf{op}(x_1, \ldots, x_n, S_1, \ldots, S_m)]\!] = \mathsf{op}(x_1, \ldots, x_n, [\![S_1]\!], \ldots, [\![S_m]\!])$ holds for all $x_1, \ldots, x_n \in \mathscr{N}$ and all $S_1, \ldots, S_m \in \mathscr{P}_S$. Of course, homomorphy requires that the respective operator is part of the source and the target language. Because of this, homomorphy is often required only for specific common operators such as the parallel operator, because it occurs more or less with the same meaning and comparable syntax in most of the process calculi. If we assume that the parallel operator is a binary operator in the source and the target language, the homomorphic translation of the parallel operator

means $[\![S_1 \mid S_2]\!] = [\![S_1]\!] \mid [\![S_2]\!]$ for all $S_1, S_2 \in \mathscr{P}_S$, where $\mid$ is the syntactical representation of the parallel operator. If single names are translated into single names, it makes sense to extend the notion of homomorphic translation to the use of a renaming policy. Thus, if the encoding translates an operator $\mathrm{op}(x_1, \ldots, x_n, S_1, \ldots, S_m)$ always into $\mathrm{op}(\phi(x_1), \ldots, \phi(x_n), [\![S_1]\!], \ldots, [\![S_m]\!])$, we call the translation of this operator again homomorphic.

Homomorphic translations of operators are e.g. used to analyse the expressive power of a single operator. To show for instance that a set of operators is not minimal the existence of an encoding is analysed that translates all operators homomorphically except for the operator that should be removed. Moreover, homomorphy is a very nice property, because it significantly eases the proof of the correctness of the encoding function. Basically, the homomorphic translation of an operator ensures that for this operator nothing is to show, because in this point the encoding obviously preserves and reflects all properties of that operator. However, even in case the respective operator is part of the source as well as the target language, homomorphy is a very strict requirement. Intuitively, it states that the encoding function is not allowed to touch the respective operator and, hence, is not allowed to simulate its behaviour by some protocol. Such translations are possible only if the compared languages are very close (at least with respect to this operator).

[43, 22, 49] show that not even between calculi that are so close as the full π-calculus (with mixed choice) and its subcalculus with only separate choice an encoding that translates the parallel operator homomorphically exists. Instead, often compositionality is required. Intuitively, an encoding is compositional if the translation of an operator is the same for all occurrences of that operator in a term. Hence, the translation of that operator can be captured by a context that is allowed in [22] to be parametrised on the free names of the respective source term.

Definition 3.3 (Compositionality) *The encoding* $[\![\cdot]\!]$ *is* compositional *if, for every operator* $\mathrm{op} : \mathscr{N}^n \times \mathscr{P}_S^m \to \mathscr{P}_S$ *and for every subset of names* N*, there exists a context* $\mathscr{C}_{\mathrm{op}}^N([\cdot]_1, \ldots, [\cdot]_{n+m}) : \mathscr{N}^n \times \mathscr{P}_S^m \to \mathscr{P}_T$ *such that, for all* $x_1, \ldots, x_n \in \mathscr{N}$ *and all* $S_1, \ldots, S_m \in \mathscr{P}_S$ *with* $\mathrm{fn}(S_1) \cup \ldots \cup \mathrm{fn}(S_m) = N$*, it holds that* $[\![\mathrm{op}(x_1, \ldots, x_n, S_1, \ldots, S_m)]\!] = \mathscr{C}_{\mathrm{op}}^N(x_1, \ldots, x_n, [\![S_1]\!], \ldots, [\![S_m]\!])$.

It does not impose additional restrictions on the introduced context. For many encodings the parameter N is not relevant or can be omitted by using instead a renaming policy on the source term names in the generated context. In contrast to homomorphy, compositionality is a very natural requirement. Intuitively, it states that every occurrence of an operator in the source term is treated by the encoding function in exactly the same way, i.e., is translated into the same term modulo the translation of the respective subterms. Also note that a compositional encoding, i.e., an encoding that translates all source term operators compositionally, implies that also any source term context can be represented as a context in the target language [45]. Moreover, compositionality guides the design of encodings, because it describes how an encoding has to look like.

Another structural criterion is the *preservation of substitutions*, denoted as *name invariance* in [22] and as *stability* in [13]. It usually requires that, for all source terms $S \in \mathscr{P}_S$ and all substitutions σ on source terms, there exists some substitution σ' on target terms such that $[\![\sigma(S)]\!] \asymp_T \sigma'([\![S]\!])$ for some equivalence $\asymp_T \subseteq \mathscr{P}_T \times \mathscr{P}_T$ on target terms. Often additional requirements on the relationship between σ and σ' or on the equivalence $\asymp_T$ are stated. The strictest case is of course that $\sigma = \sigma'$ and $\asymp_T = \equiv_\alpha$. This criterion is based on the idea that names are property-less [13]. Hence, the preservation of substitutions should ensure that encodings of source terms that differ only in their free names can also only differ in free names (modulo the provided equivalence).

In [22] name invariance is defined modulo the introduced renaming policy. Accordingly, an encoding is considered independent of specific names if it preserves all substitutions σ on source terms by a substitution σ' on target terms such that σ' respects the changes made by the renaming policy.

Definition 3.4 (Name Invariance) *The encoding* $[\![\cdot]\!]$ *is* name invariant *if, for every* $S \in \mathscr{P}_S$ *and* σ*, it holds that*

$$[\![\sigma(S)]\!] \begin{cases} \equiv_\alpha \sigma'([\![S]\!]) & \text{if } \sigma \text{ is injective} \\ \asymp_T \sigma'([\![S]\!]) & \text{otherwise} \end{cases}$$

where σ' *is such that* $\phi(\sigma(n)) = \sigma'(\phi(n))$ *for every* $n \in \mathcal{N}$.

Moreover, [59] present *link independence*, a condition that prevents encodings from introducing free names. More precisely, link independence means that, for all source terms $S_1, S_2 \in \mathscr{P}_S$, $\mathsf{fn}(S_1) \cap \mathsf{fn}(S_2) = \emptyset$ implies $\mathsf{fn}([\![S_1]\!]) \cap \mathsf{fn}([\![S_2]\!]) = \emptyset$.

We denote the criteria presented in this subsection as *structural criteria* because they focus mainly on structural properties of the encoding function. We should, however, be aware that every criterion limits the existence of encoding functions and that such limitations usually also have a semantic effect. The main purpose of compositionality is to ensure that the encoding function is compositional. But compositionality also forbids for global coordination (see Section 3.6). Because of that, there is a number of well-accepted encodings—as e.g. the encoding of the π-calculus into the join-calculus in [11]—that are not compositional. Instead the encoding in [11] consists of two levels: an outer level that is parametrised on the free names of the source term and an inner compositional encoding.

3.6 Domain-Specific Criteria

Above we presented different kinds of criteria that were used to specify the notion of a "good" encoding in the literature. Thereby the purpose of all criteria introduced so far, is to define a general notion of correctness for encodings. We know that there is no agreement about the choice, combination, or concrete variant of the above criteria for a general notion of correctness or quality of an encoding. But even if we would agree on such a general notion of quality, there would still be the need for domain-specific criteria. A general notion is in particular necessary, if we want to compare different results or build a hierarchy. Accordingly, such a general notion of quality is a good starting point for language comparison. Nonetheless, we will sometimes need to extend it by domain-specific criteria. Domain-specific criteria, as the name suggests, are used to analyse properties of a specific domain that may in general not be interesting. Hence, it is not a good idea to overload a general framework by permanently adding domain-specific criteria. Instead, we add a domain-specific criterion only if it is necessary to answer a particular kind of question. Possible domains in the context of process calculi are e.g. causality, the branching time behaviour of processes, or considerations related to some special features as failures or time constrains.

Note that an additional criterion may strengthen an encodability result, but it weakens separation results. Moreover, already a single additional criterion significantly complicates the comparison of a result with other already established results that do not rely on this additional criterion. Quite often, they even lead to incomparable results; in particular if two results use different additional criteria. Hence, domain-specific criteria should only be used if they are unavoidable to answer a specific kind of question.

As an example for a domain-specific criterion, we discuss how to obtain a criterion that ensures that an encoding preserves the degree of distribution. Given an extension of a process calculus with an explicit notion of distribution or location we can define the degree of distribution of a system as the number of its locations. If locations are not defined explicitly, a process P is distributable into $P_1, \ldots, P_n$, if we find some distribution that extracts $P_1, \ldots, P_n$ from within P onto different locations. Preservation of distribution then means that the target term is as least as distributable as the source

term. Note that the operator of process calculi that is usually associated with distribution is the parallel operator. Accordingly, we consider components of a term that are composed in parallel as distributable. More precisely, we understand distribution as the separation of a process into its (sequential) components.

Hence, by studying distribution preserving encodings, we analyse the possibilities to implement the operators of a calculus or especially its parallel operator. If it is always possible to preserve the degree of distribution in an encoding of a source language into a target language which is close to an implementation e.g. in a real world scenario, then the corresponding parallel operator can be implemented in this scenario simply as the operator of distribution, i.e., parallel source terms can be implemented in distributed real world processes. If it is not possible to obtain a distribution preserving encoding, then the source language implicitly defines side conditions on the use of the parallel operator usually induced by the defined synchronisation mechanism that forbids for such simple implementations. Thus, the implementation of parallel source terms as distributed processes may be possible only under some side conditions, which are hopefully already paraphrased by the respective separation result. In particular, an encoding that preserves the degree of distribution should not introduce a coordinator for concurrent actions, because if concurrent actions are coordinated by the same instance they are sequentialised, i.e., the implementation is less efficient.

Note that compositionality in Definition 3.3 already prevents from the use of global coordinators. Compositionality requires that all occurrences of a parallel operator have to be translated basically in the same way. Hence, if such an encoding introduces a coordinator then for every parallel operator a coordinator is introduced and there is no possibility to examine which of them is the outermost or to order them, i.e., it is not possible to coordinate the coordinators such that they proceed as a centralised entity. In that view, compositionality can be seen as a minimal criterion to ensure the preservation of distribution. However, compositionality alone is too weak, because it still allows for *local coordinators*, i.e., a compositional encoding may still sequentialise some parts of a source term (see e.g. the encodings in [52, 24]).

Instead the homomorphic translation of the parallel operator, i.e., $[\![P \mid Q]\!] = [\![P]\!] \mid [\![Q]\!]$, was often used as a criterion to measure whether an encoding respects the degree of distribution (see e.g. [43, 10, 30]). The homomorphic translation of the parallel operator forbids the introduction of (global and local) coordinators for the translation of the parallel operator. As discussed in [49, 52, 53] the homomorphic translation of the parallel operator usually implies that the respective encoding indeed preserves the degree of distribution but that the converse is not true, i.e., there are encodings that do not translate the parallel operator homomorphically but preserve nonetheless the degree of distribution of all source terms. In this sense, the homomorphic translation of the parallel operator is too strict—at least for separation results. It rightly forbids the introduction of coordinators that reduce the degree of distribution. But it also forbids protocols that handle communications of parallel components without sequentialising them or reducing the degree of distribution in another sense. Moreover, the homomorphic translation of the parallel operator is not always suited to reason about distribution in process calculi. For example in the join-calculus it is not always possible to separate distributed components by means of a parallel operator.

[49, 52, 53] introduce an alternative criterion for the preservation of the degree of distribution. Compositionality and the homomorphic translation of the parallel operator are both structural criteria. Hence, one may assume that also the preservation of distribution is a structural criterion, but in fact this is not true. A natural first condition is to require that encoded source terms are at least as distributable as the source term itself, i.e., that the degree of distribution has to be preserved by the encoding. However, it does not suffice to reason about the degree of distribution, i.e., about the number of distributable components, without additional requirements on the components. An encoding can always trivially ensure that the encoding has at least as much distributable components by introducing new components without any

behaviour. Thus, we require that the encodings of distributable source term parts and their correspond-
ing parts in the encoding are related by $\asymp_T$. By doing so we relate the definition of the preservation of
distributability to operational completeness, i.e., a semantic criterion that ensures the preservation of the
behaviour of the source term (part). Hence, we require that each target term part can emulate at least all
behaviour of the respective source part. As a side effect, we require, that whenever a part of a source
term can solve a task independently of the other parts, i.e., it can reduce on its own, then the respective
part of its encoding must also be able to emulate this reduction independently of the rest of the encoded
term. This reflects our intuition that distribution adds some additional requirements on the independence
of parallel terms. Accordingly, we require that not only the source term and its encoding are distributable
to the same degree, but also their derivatives, i.e., we do not only consider the *initial* degree of distribu-
tion. Because of that, the criterion that is presented in [53] has both a structural as well as a semantic
component. Remember that a term P is distributable into $P_1, \ldots, P_n$, if we find some distribution that
extracts $P_1, \ldots, P_n$ from within P onto different locations.

Definition 3.5 (Preservation of Distribution) *An encoding* $[\![\cdot]\!] : \mathscr{P}_S \to \mathscr{P}_T$ *preserves distribution if for
every* $S \in \mathscr{P}_S$ *and for all terms* $S_1, \ldots, S_n \in \mathscr{P}_S$ *that are distributable within S there are some* $T_1, \ldots, T_n \in
\mathscr{P}_T$ *that are distributable within* $[\![S]\!]$ *such that* $T_i \asymp_T [\![S_i]\!]$ *for all* $1 \leq i \leq n$.

   In essence, this requirement is a distributability-enhanced adaptation of operational completeness.
Whenever a source term is distributable into n terms then its encoding must again be distributable into
n terms, i.e., the encoded source term is at least as distributable as the source term itself. Moreover, if
some of these n terms, say S_i, can perform some execution independently of the rest then, by operational
completeness, this execution has to be emulated by its translation modulo $\asymp_T$, i.e., $S_i \Longmapsto_S S_i'$ implies
$T_i \Longmapsto_T \asymp_T [\![S_i']\!]$. This formalisation of the preservation of distributability respects both the intuition on
distribution as separation on different locations—captured by the structural requirement that the encoded
source term is at least as distributable as the source term itself—as well as the intuition on distribution as
independence of processes and their executions—a semantic requirement implemented by the condition
$T_i \asymp_T [\![S_i]\!]$.
   The main disadvantage of this criterion is its complexity. It is more model-independent, since it does
not rely on the notion of the parallel operator, i.e., is applicable also for calculi without a parallel operator
or completely different kinds of parallel operators such as in CSP and it can be applied to calculi like the
join-calculus, where distributed components are not always separated by a parallel operator. But, in order
to apply this criterion, we have to specify in the source and the target language what it means for a process
to be distributable into a set of components. In calculi with explicit locations this is usually easy. But it
is difficult to find a general, i.e., model-independent, formalisation of distributability (see [49, 52, 53]).
Moreover, this criterion was designed in order to be combined with operational correspondence and
is strongly connected to this criterion. The advantage of the homomorphic translation of the parallel
operator is that it is simple, easy to understand, and described independently of other criteria. But it is
too strong for separation results, whereas compositionality is too weak for encodability results.

3.7   Summary

We introduced a number of encodability criteria. We observe that these criteria appear in different vari-
ants and that there is no agreement on a set of criteria that should be used; not even if we ignore domain-
specific criteria. Moreover, we observe that some criteria are often used but as the discussion on full
abstraction (see Section 3.2 and [13, 23, 46]) shows not always well-understood. For domain-specific

criteria the situation is usually worse, i.e., it is even more difficult to analyse whether they are suitable. A mechanism to reason about the quality of encodability criteria is discussed in Section 5.

Encodability criteria define properties of the encoding function and should ensure that an encoding can be considered as meaningful. Unfortunately, encodability and separation results that are derived w.r.t. different sets of encodability criteria or different variants of these criteria are hard to compare. Section 4 analyses general frameworks of encodability criteria. Note that the formalisation of a criterion should be as general as possible, because a formalisation that is to close to a specific process calculus may hinder the derivation of similar results for other calculi and, thus, the comparison with other results. Furthermore, it is sometimes easier to define a new criterion with respect to an existing one, but again this may shrink the possibilities to compare to other results that do not satisfy the old criterion.

Finally, note that the criteria pose different kinds of proof obligations on the correctness proofs of encodings. Structural criteria (and also the criteria for the efficiency of an encoding function) are usually easier to verify than the remaining semantic criteria. But the semantic criteria are often considered as more essential. Unfortunately, we are not only lacking mechanisms to analyse the quality of encodability criteria and an agreement of a general notion of a "good" encoding but also general proof techniques for encodability criteria. From the criteria introduced above full abstraction and operational correspondence are usually the most elaborate criteria to prove. In the case of full abstraction the proofs heavily depend on the chosen pair of equivalences. Operational correspondence is usually shown by induction on the nature of source or target term steps. But some more sophisticated study of proof techniques might provide e.g. some hints on how to deal with the intermediate states that weakly operationally corresponding encodings might introduce (compare e.g. to the discussion of pre- and post-processing steps in [49]).

4   A General Notion of Quality

A general notion of quality is important to reason about the general expressive power of languages and to compare different results. In particular, they are essential to build a hierarchy that can only be based on encodability and separation results w.r.t. the same set of criteria. Moreover, if we want to relate the expressive power of two given languages there is no guidance on how to start or whether an obtained result is sufficiently substantiated by the chosen criteria to call it reasonable. There are basically two problems that we have to face in providing such a general notion: (1) We have to choose the nature of criteria. For a general setting the criteria should be model-independent. This also includes to some extent the relations that are used e.g. in full abstraction and operational correspondence, i.e., a general setting has to specify how to choose them. Moreover, the criteria should be designed to capture properties of the encoding that are generally agreed as being useful. (2) We have to decide on the variants of the respective criteria. Weaker criteria allow for more encodability and less separation results, whereas more restrictive criteria allow for less encodability and more separation results. The difficulty is to identify the sweet spot between too restrictive and too weak criteria such that the variants are meaningful for separation as well as encodability results. As discussed in Section 3.6, there is a need for domain-specific criteria. Accordingly, it should be possible to extend a general framework by domain-specific criteria.

4.1   Towards A Unified Approach to Encodability and Separation Results

In order to provide a general framework, [19, 22] suggests five criteria well suited for language comparison, i.e., for positive as well as negative translational results. As claimed in [22], most of the encodings appearing in the literature satisfy this framework and several known separation results can also be de-

rived within this framework but there are also encodings that do not satisfies this framework, i.e., the framework is not trivial. The set of criteria is small and handy but at the same time guides the design of encoding functions and supports the proof of translational results by separating the requirements on different intuitive criteria. This framework specifies an encoding to be "good" if it satisfies the five presented criteria. In [19, 20, 21, 22] a number of encodability and separation results—including some new results—that are derived in this setting can be found.

The five conditions are divided into two structural and three semantic criteria. The structural criteria include (1) *compositionality* and (2) *name invariance*. The semantic criteria include (3) *operational correspondence*, (4) *divergence reflection*, and (5) *success sensitiveness*. Note that for the definition of name invariance and operational correspondence a behavioural equivalence $\asymp_T$ for the target language is assumed. Its purpose is to describe the abstract behaviour of a target process, where abstract refers to the behaviour of the source term.

The two structural criteria were introduced and discussed in Section 3.5. Compositionality is given in Definition 3.3 and name invariance in Definition 3.4. They state that the encoding function should be compositional and independent of specific names. To ensure that there are no conflicts between names that are introduced by the encoding function for technical reasons, i.e., to implement some kind of protocol, and the source term names, the encoding is equipped with a renaming policy (Definition 2.3).

The first semantic criterion and usually the most elaborate one to prove is weak operational correspondence as given in Definition 3.1. The definition of operational correspondence relies on the equivalence $\asymp_T$ to get rid of junks possibly left over within computations of target terms. To deal with infinite computations in encodings, the second semantic criterion requires that the encoding reflects divergence. It ensures that the encoding function cannot introduce new divergent behaviour, i.e., all divergent target terms are due to the encoding of a divergent source term. The last criterion links the behaviour of source terms to the behaviour of their encodings. [22] assumes a *success* operator ✓ as part of the syntax of both the source and the target language. An encoding preserves the abstract behaviour of the source term if it and its encoding answer the tests for success in exactly the same way (Definition 3.2). This criterion only links the behaviours of source terms and their literal translations, but not of their derivatives. To do so, [22] relates success sensitiveness and operational correspondence by requiring that the equivalence on the target language never relates two processes with different success behaviours.

Definition 4.1 (Success Respecting) *An equivalence $\mathscr{R}$ is* success respecting *if, for every $P, Q \in \mathscr{P}_C$ with $P \Downarrow_\checkmark$ and $Q \not\Downarrow_\checkmark$, it holds that $(P, Q) \notin \mathscr{R}$. We require that $\asymp_T$ is a success respecting equivalence.*

The combination of success sensitiveness and operational correspondence allows to compare the behaviour of source and target terms even if they do not share common observables. By [22] a "good" equivalence $\asymp_T$ is often defined in the form of a barbed equivalence (as described e.g. in [37]) or can be derived directly from the reduction semantics (as described e.g. in [28]) and is often a congruence, at least with respect to parallel composition.

4.2 Theory of Interaction

In contrast to [22], the general framework in [13] is based on labelled semantics. Intuitively, they combine full abstraction and operational correspondence into a bisimulation-like relation called *subbisimulation*. Subbisimulation in [13] connects labelled executions of the source and the target language. Moreover, preservation and reflection of divergence is required. The approach is then used to compare different variants of CCS and different variants of the π-calculus. For example subbisimilarity is applied to show the independence of the operators of the π-calculus.

The paper [12] extends this approach into a general theory of interaction. The prime motivation for the theory of interaction is to bridge the gap between the computation theory, i.e., what kind of functions can be computed, and the interaction theory. Therefore four fundamental and model-independent principles are presented and from them a general theory of equality and expressiveness are derived. Again subbisimilarity but without labels is used as criterion for encodings.

Definition 4.2 (Subbisimilarity) *A relation $\mathscr{R}$ is a* subbisimilarity *if it is total, sound, equipollent, extensional, codivergent, and bisimilar.*

Intuitively, (1) total means that for every source term there is a target term, (2) soundness is similar to operational soundness, (3) equipollence implies that related terms either both cannot reach any state with an observable or both can reach a state with (potentially different) observables, (4) extensional means congruent w.r.t. the parallel operator and restriction, (5) codivergent means that the relation is divergence respecting, and (6) bisimilarity is defined similar to Definition 2.1.

In comparison to the framework of Gorla the requirements induced by the formulation of subbisimilarity seem to be stricter, although a direct comparison is difficult, because of the different formulations. However, [13] and [12] make use of a stricter variant of operational correspondence that does not allow for intermediate or partially committed states. Moreover, [13, 12] fix a number of assumptions on process calculi, e.g. that all process calculi contain at least the parallel operator and restriction.

4.3   Musings on Encodings and Expressiveness

Also [17, 18] aims at providing a general notion of expressiveness for language comparison. Here an encoding should be *valid* and *correct*. In [18] these concepts are defined up to a semantic equivalence or preorder $\sim$, that is not fixed by the framework.

Intuitively, an encoding is valid if it preserves the meaning of expressions, i.e., such that the meaning of a translated expression is semantically equivalent to the meaning of the original. Therefore, [18] assumes a function for each language that associates the *meaning* to processes by mapping them on values (for simplicity and in contrast to [18] I ignore in the following that processes may contain variables). Different languages may have different domains of values. The semantic relation $\sim$ is used to mediate between these different domains, i.e., it is assumed that it is a relation on values over a universe that contains the domains of the source as well as the target language. Moreover, [18] requires a *semantic translation* **R** that is a relation that relates each value of the source with its counterpart (or counterparts) in the target. Then, an encoding is *correct* iff the meaning of the translation of an expression is a counterpart of the meaning of this expression.

Definition 4.3 (Correctness) *An encoding $[\![\cdot]\!]$ is* correct *w.r.t. a semantic translation* **R** *if* **R** *relates the meaning of S and $[\![S]\!]$ for all $S \in \mathscr{P}_{\mathrm{S}}$.*

An encoding $[\![\cdot]\!]$ is correct *up to $\sim$ iff $\sim$ is an equivalence, the restriction* **R** *of $\sim$ to the cross product of the domain of the source and the target is a semantic translation, and $[\![\cdot]\!]$ is correct w.r.t.* **R**.

Definition 4.4 (Validity) *An encoding $[\![\cdot]\!]$ is* valid *up to $\sim$ iff it is correct w.r.t. some semantic translation* **R** $\subseteq \sim$. *Language $\mathscr{L}_{\mathrm{T}}$ is at least as expressive as $\mathscr{L}_{\mathrm{S}}$ up to $\sim$ if an encoding valid up to $\sim$ from $\mathscr{L}_{\mathrm{S}}$ into $\mathscr{L}_{\mathrm{T}}$ exists.*

As discussed in [18], in comparison to [22] the above approach implies a slightly stricter variant of compositionality (without the parameter on a set of names N) but no name invariance criterion. Moreover, the combination of three semantic criteria of [22] are close to an instantiation of the criteria in this approach with a particular preorder $\sim$, namely a success respecting and divergence respecting variant of bisimulation (see Section 5).

5　Formalising and Analysing Encodability Criteria

There exists a bunch of different criteria and different variants of criteria in order to reason in different settings. This leads to incomparable results. Moreover it is not always clear whether the criteria used to obtain a result in a particular setting do indeed fit to this setting. A way to formally reason about and compare encodability criteria is presented in [50]. The main idea of this approach is to map encodability criteria on requirements on a relation between source and target terms that is induced by the encoding function. This way the problem of analysing or comparing encodability criteria is reduced to the better understood problem of comparing relations on processes. An Isabelle/HOL formalisation can be found in [51].

The different purposes of encodability criteria lead to very different kinds of conditions that are usually hard to analyse and compare directly. In fact even widely used criteria—as full abstraction— seem not to be fully understood by the community, as the need for articles as [23, 46] shows. In contrast to that, relations on processes—such as simulations and bisimulations—are a very well studied and understood topic (see for example [16]). Moreover, it is natural to describe the behaviour of terms, or compare them, modulo some equivalence relation. Also many encodability criteria, like operational correspondence, are obviously designed with a particular kind of relation between processes in mind. Therefore, in order to be able to formally reason about encodability criteria, to completely capture and describe their semantic effect, and to analyse side conditions of combinations of criteria, mapping them on conditions on relations between source and target terms seems to be natural.

According to [50] every encoding $[\![\cdot]\!] : \mathscr{P}_S \to \mathscr{P}_T$ induces a relation $\mathscr{R}_{[\![\cdot]\!]} \subseteq (\mathscr{P}_S \uplus \mathscr{P}_T)^2$ on the disjoint union of its source and target terms by relating source terms and their literal translations, i.e., $(S, [\![S]\!]) \in \mathscr{R}_{[\![\cdot]\!]}$ for all $S \in \mathscr{P}_S$. Encodability criteria induce further properties on such relations. By analysing the different kinds of such properties the semantic effect of an encodability criterion can be studied and different criteria can be compared in a model-independent and formal way. In order to completely capture the effect of a criterion, [50] aims at iff-results of the form

> $[\![\cdot]\!]$ satisfies **C** iff there exists a relation $\mathscr{R}_{[\![\cdot]\!]}$ such that $\forall S.\ (S, [\![S]\!]) \in \mathscr{R}_{[\![\cdot]\!]}$ and $\mathsf{P}\big(\mathscr{R}_{[\![\cdot]\!]}\big)$,

where P is the condition that captures the effect of **C**. For example, an encoding reflects divergence iff there exists a relation $\mathscr{R}_{[\![\cdot]\!]}$ such that $\forall S.\ (S, [\![S]\!]) \in \mathscr{R}_{[\![\cdot]\!]}$ and $\mathscr{R}_{[\![\cdot]\!]}$ reflects divergence. Similarly, an encoding (weakly/strongly) respects barbs iff there exists a relation $\mathscr{R}_{[\![\cdot]\!]}$ such that $\forall S.\ (S, [\![S]\!]) \in \mathscr{R}_{[\![\cdot]\!]}$ and $\mathscr{R}_{[\![\cdot]\!]}$ (weakly/strongly) respects barbs.

Using this technique, [50] shows that without further requirements on the source and target relations $\mathscr{R}_S$ and $\mathscr{R}_T$, that are used in the formulation of full abstraction, the semantic effect of full abstraction is very small. Full abstraction is mapped on a relation that relates at least each source term to its literal translation and includes the relations $\mathscr{R}_S$ and $\mathscr{R}_T$. Let us additionally add pairs of the form $([\![S]\!], S)$ for all $S \in \mathscr{P}_S$. Then, an encoding is fully abstract w.r.t. the preorders $\mathscr{R}_S$ and $\mathscr{R}_T$ iff there exists a transitive relation $\mathscr{R}_{[\![\cdot]\!]}$ that relates at least each source term to its literal translation in both directions, such that the restriction of $\mathscr{R}_{[\![\cdot]\!]}$ to source terms $\mathscr{R}_{[\![\cdot]\!]}\!\restriction_{\mathscr{P}_S}$ is $\mathscr{R}_S$ and $\mathscr{R}_{[\![\cdot]\!]}\!\restriction_{\mathscr{P}_T} = \mathscr{R}_T$.

Lemma 5.1 (Full Abstraction, [50]) $[\![\cdot]\!] : \mathscr{P}_S \to \mathscr{P}_T$ *is* fully abstract *w.r.t. the preorders* $\mathscr{R}_S \subseteq \mathscr{P}_S^2$ *and* $\mathscr{R}_T \subseteq \mathscr{P}_T^2$ *iff* $\exists \mathscr{R}_{[\![\cdot]\!]}.\ \big(\forall S.\ (S, [\![S]\!]), ([\![S]\!], S) \in \mathscr{R}_{[\![\cdot]\!]}\big) \wedge \mathscr{R}_S = \mathscr{R}_{[\![\cdot]\!]}\!\restriction_{\mathscr{P}_S} \wedge \mathscr{R}_T = \mathscr{R}_{[\![\cdot]\!]}\!\restriction_{\mathscr{P}_T} \wedge \mathscr{R}_{[\![\cdot]\!]}$ *is transitive.*

Thus an encoding is fully abstract w.r.t. $\mathscr{R}_S$ and $\mathscr{R}_T$ if the encoding function combines the relations $\mathscr{R}_S$ and $\mathscr{R}_T$ in a transitive way. This underpins the discussions in [13, 23, 46] showing that full abstraction without a clarification of the respective equivalences is not meaningful. It also shows the need for a formal analysis of encodability criteria.

The formulation of operational correspondence (in all its variants) strongly reminds us of simulation relations on processes, such as bisimilarity. Obviously this criterion is designed in order to establish a simulation-like relation between source and target terms. By mapping this criterion on properties of a relation, we can determine the exact nature of this relation. The first two variants of Definition 3.1 exactly describe strong and weak bisimilarity up to $\asymp_T$.

Lemma 5.2 (Operational Correspondence, [50]) *An encoding* $[\![\cdot]\!] : \mathscr{P}_S \to \mathscr{P}_T$ *is operational corresponding w.r.t. a preorder* $\asymp_T \subseteq \mathscr{P}_T^2$ *that is a bisimulation iff* $\exists \mathscr{R}_{[\![\cdot]\!]}. \ \big(\forall S. \ (S, [\![S]\!]) \in \mathscr{R}_{[\![\cdot]\!]} \big) \wedge \asymp_T = \mathscr{R}_{[\![\cdot]\!]} {\upharpoonright} \mathscr{P}_T \wedge \big(\forall S, T. \ (S, T) \in \mathscr{R}_{[\![\cdot]\!]} \ implies \ ([\![S]\!], T) \in \asymp_T \big) \wedge \mathscr{R}_{[\![\cdot]\!]} \ is \ a \ preorder \ and \ a \ bisimulation.$

Here, the conditions $\mathscr{R}_T = \mathscr{R}_{[\![\cdot]\!]} {\upharpoonright} \mathscr{P}_T$ and $\big(\forall S, T. \ (S, T) \in \mathscr{R}_{[\![\cdot]\!]} \to ([\![S]\!], T) \in \mathscr{R}_T \big)$ are technical side conditions for the proof of the only-if-part that ensure that $\asymp_T$ is a bisimulation. We obtain the same result if we replace operational correspondence by strong operational correspondence and bisimulation by strong bisimulation. Accordingly, operational correspondence ensures that source terms and their translations are reduction bisimilar.

To obtain a similar result for weak operational correspondence, [50] had to introduce a new kind of simulation relation denoted as correspondence simulation.

Lemma 5.3 (Weak Operational Correspondence, [50]) $[\![\cdot]\!]$ *is weakly operational corresponding w.r.t. a preorder* $\asymp_T \subseteq \mathscr{P}_T^2$ *that is a correspondence simulation iff* $\exists \mathscr{R}_{[\![\cdot]\!]}. \ \big(\forall S. \ (S, [\![S]\!]) \in \mathscr{R}_{[\![\cdot]\!]} \big) \wedge \asymp_T = \mathscr{R}_{[\![\cdot]\!]} {\upharpoonright} \mathscr{P}_T \wedge \big(\forall S, T. \ (S, T) \in \mathscr{R}_{[\![\cdot]\!]} \ implies \ ([\![S]\!], T) \in \asymp_T \big) \wedge \mathscr{R}_{[\![\cdot]\!]} \ is \ a \ preorder \ and \ a \ correspondence \ simulation.$

We omit the definition of correspondence simulation but point out that it is a simulation relation that is in between coupled similarity (Definition 2.2) and bisimulation. Accordingly, weak operational correspondence ensures that source terms and their literal translations are coupled similar.

As stated in [50], the combination of the above results implies that the three semantic criteria of [22] ensure that any "good" encoding in this framework relates source and target terms by a coupled simulation that reflects divergence and respects success. The approach presented in [13, 12] was not addressed in [50] but it requires itself a simulation relation between source and target terms. Subbisimilarity is a variant of bisimulation. Accordingly, the requirements of [13, 12] are more restrictive than [22]. The analysis of structural criteria is left for further research, because structural criteria need more assumptions on the considered languages. As discussed in [50], the formal analysis of encodability criteria can also help to derive proof methods for the respective criteria.

6 Conclusions

As stated in the end of Section 3, there are basically three lines of further research: (1) We need techniques to reason about the quality of encodability criteria to ensure that our encodings are indeed meaningful. Section 5 revises one way to do that, but raises itself some open questions as e.g. how to deal with structural criteria. (2) In order to compare results and build hierarchies we need a general notion of the quality of an encoding. Section 4 outlines three different frameworks for this purpose. All three approaches have certain basic ideas in common, e.g. all three variants imply structural and semantic criteria. But the existence of three frameworks shows that there is still no general agreement and indeed these three frameworks differ not only w.r.t. the semantic equivalence they induce (see Section 5) they also use quite different terminologies and basic definitions. With that they impose different proof techniques. (3) Besides the analysis of encodability criteria, we are also lacking a study of proof techniques for particular criteria.

References

[1] S. Arun-Kummar & M. Hennessy (1992): *An efficiency preorder for processes*. Acta Informatica 29(8), pp. 737–760, doi:10.1007/BF01191894.

[2] J.C.M. Baeten (2005): *A brief history of process algebra*. Theoretical Computer Science 335(2–3), pp. 131–146, doi:10.1016/j.tcs.2004.07.036.

[3] M. Baldamus, J. Parrow & B. Victor (2005): *A Fully Abstract Encoding of the π-Calculus with Data Terms (Extended Abstract)*. In: Proc. of ICALP, LNCS 3580, Springer, pp. 1202–1213, doi:10.1007/11523468_97.

[4] J.A. Bergstra & J.W. Klop (1982): *Fixed point semantics in process algebra*. Technical Report IW 206/82, Mathematical Centre, Amsterdam.

[5] F.S. Boer & C. Palamidessi (1991): *Embedding as a tool for Language Comparison: On the CSP hierarchy*. In: Proc. of CONCUR, LNCS 527, Springer, pp. 127–141, doi:10.1007/3-540-54430-5_85.

[6] G. Boudol (1992): *Asynchrony and the π-calculus (note)*. Note, INRIA.

[7] G.N. Buckley & A. Silberschatz (1983): *An Effective Implementation for the Generalized Input-Output Construct of CSP*. ACM Transactions on Programming Languages and Systems (TOPLAS) 5(2), pp. 223–235, doi:10.1145/69624.357208.

[8] N. Busi, M. Gabbrielli & G. Zavattaro (2009): *On the expressive power of recursion, replication and iteration in process calculi*. Mathematical Structures in Computer Science 19(6), pp. 1191–1222, doi:10.1017/S096012950999017X.

[9] D. Cacciagrano, F. Corradini, J. Aranda & F.D. Valencia (2008): *Linearity, Persistence and Testing Semantics in the Asynchronous Pi-Calculus*. Electronic Notes in Theoretical Computer Science 194(2), pp. 59–84, doi:10.1016/j.entcs.2007.11.006.

[10] M. Carbone & S. Maffeis (2003): *On the Expressive Power of Polyadic Synchronisation in π-Calculus*. Nordic Journal of Computing 10(2), pp. 70–98, doi:10.1016/S1571-0661(05)80361-5.

[11] C. Fournet & G. Gonthier (1996): *The Reflexive CHAM and the Join-Calculus*. In: Proc. of POPL, SIGPLAN-SIGACT, ACM, pp. 372–385, doi:10.1145/237721.237805.

[12] Y. Fu (2016): *Theory of Interaction*. Theoretical Computer Science 611, pp. 1–49, doi:10.1016/j.tcs.2015.07.043.

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

[14] R.J. van Glabbeek (1993): *The Linear Time – Branching Time Spectrum II*. In: Proc. of CONCUR, LNCS 715, pp. 66–81, doi:10.1007/3-540-57208-2_6.

[15] R.J. van Glabbeek (1994): *On the expressiveness of ACP (extended abstract)*. In: Proc. of ACP, Workshops in Computing, pp. 188–217, doi:10.1007/978-1-4471-2120-6_8.

[16] R.J. van Glabbeek (2001): *The Linear Time – Branching Time Spectrum I: The Semantics of Conrete, Sequential Processes*. Handbook of Process Algebra, pp. 3–99, doi:10.1016/B978-044482830-9/50019-9.

[17] R.J. van Glabbeek (2012): *Musings on Encodings and Expressiveness*. In: Proc. of EXPRESS/SOS, EPTCS 89, pp. 81–98, doi:10.4204/EPTCS.89.7.

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

[19] D. Gorla (2008): *Towards a Unified Approach to Encodability and Separation Results for Process Calculi*. In: Proc. of CONCUR, LNCS 5201, pp. 492–507, doi:10.1007/978-3-540-85361-9_38.

[20] D. Gorla (2009): *On the Relative Expressive Power of Calculi for Mobility*. In: Proc. of MFPS, ENTCS 249, pp. 269–286, doi:10.1016/j.entcs.2009.07.094.

[21] 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.

[22] 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.

[23] D. Gorla & U. Nestmann (2014): *Full abstraction for expressiveness: history, myths and facts*. *Mathematical Structures in Computer Science*, pp. 1–16, doi:10.1017/S0960129514000279.

[24] M. Hatzel, C. Wagner, K. Peters & U. Nestmann (2015): *Encoding CSP into CCS*. In: *Proc. of EXPRESS/SOS*, *EPTCS* 7, pp. 61–75, doi:10.4204/EPTCS.190.5.

[25] C. Hewitt, P. Bishop & R. Steiger (1973): *A universal modular ACTOR formalism for artificial intelligence*. In: *Proc. of IJCAI*, ACM, pp. 235–245.

[26] C.A.R. Hoare (1978): *Communicating Sequential Processes*. Communications of the ACM 21(8), pp. 666–677, doi:10.1145/359576.359585.

[27] K. Honda & M. Tokoro (1991): *An Object Calculus for Asynchronous Communication*. In: *Proc. of ECOOP*, *LNCS* 512, pp. 133–147, doi:10.1007/BFb0057019.

[28] K. Honda & N. Yoshida (1995): *On Reduction-Based Process Semantics*. *Theoretical Computer Science* 151(2), pp. 437–486, doi:10.1016/0304-3975(95)00074-7.

[29] F. Knabe (1993): *A Distributed Protocol for Channel-Based Communication with Choice*. *Computers and Artificial Intelligence* 12(5), pp. 475–490.

[30] C. Laneve & A. Vitale (2010): *The Expressive Power of Synchronizations*. In: *Proc. of LICS*, IEEE, pp. 382–391, doi:10.1109/LICS.2010.15.

[31] R.J. Lipton, L. Snyder & Y. Zalcstein (1974): *A Comparative Study of Models of Parallel Computation*. In: *Proc. of SWAT*, IEEE, pp. 145–155, doi:10.1109/SWAT.1974.2.

[32] R. Milner (1989): *Communication and Concurrency*. Prentice-Hall, Inc.

[33] R. Milner (1992): *Functions as Processes*. *Mathematical Structures in Computer Science* 2(2), pp. 119–141, doi:10.1017/S0960129500001407.

[34] R. Milner (1993): *The Polyadic π-Calculus: a Tutorial*. *Logic and Algebra of Specification* 94, pp. 203–246, doi:10.1007/978-3-642-58041-3_6.

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

[36] 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.

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

[38] J.C. Mitchell (1993): *On abstraction and the expressive power of programming languages*. *Science of Computer Programming* 21(2), pp. 141–163, doi:10.1016/0167-6423(93)90004-9.

[39] U. Nestmann (1996): *On Determinacy and Nondeterminacy in Concurrent Programming*. Ph.D. thesis, Universität Erlangen-Nürnberg.

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

[41] U. Nestmann (2006): *Welcome to the Jungle: A subjective Guide to Mobile Process Calculi*. In: *Proc. of CONCUR*, *LNCS* 4137, pp. 52–63, doi:10.1007/11817949_4.

[42] U. Nestmann & B.C. Pierce (2000): *Decoding Choice Encodings*. *Information and Computation* 163(1), pp. 1–59, doi:10.1006/inco.2000.2868.

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

[44] C. Palamidessi, V.A. Saraswat, F.D. Valencia & B. Victor (2006): *On the Expressiveness of Linearity vs Persistence in the Asychronous Pi-Calculus*. In: *Proc. of LICS*, IEEE Computer Society, pp. 59–68, doi:10.1109/LICS.2006.39.

[45] 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.

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

[47] J. Parrow & P. Sjödin (1992): *Multiway Synchronization Verified with Coupled Simulation*. In: *Proc. of CONCUR*, *LNCS* 630, pp. 518–533, doi:10.1007/BFb0084813.

[48] J.A. Perez (2009): *Higher-Order Concurrency: Expressiveness and Decidability Results*. Ph.d. thesis, University of Bologna.

[49] K. Peters (2012): *Translational Expressiveness*. Ph.D. thesis, TU Berlin. Available at `http://opus.kobv.de/tuberlin/volltexte/2012/3749/`.

[50] K. Peters & R. van Glabbeek (2015): *Analysing and Comparing Encodability Criteria*. In: *Proc. of EXPRESS/SOS*, *EPTCS* 190, pp. 46–60, doi:10.4204/EPTCS.190.4.

[51] K. Peters & R. van Glabbeek (2015): *Analysing and Comparing Encodability Criteria for Process Calculi*. Archive of Formal Proofs. `http://isa-afp.org/entries/Encodability_Process_Calculi.shtml`.

[52] K. Peters & U. Nestmann (2012): *Is It a "Good" Encoding of Mixed Choice?* In: *Proc. of FoSSaCS*, *LNCS* 7213, pp. 210–224, doi:10.1007/978-3-642-28729-9_14.

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

[54] C.A. Petri (1962): *Kommunikation mit Automaten*. Ph.D. thesis, Institut für Instrumentelle Mathematik, Bonn.

[55] J.G. Riecke (1991): *Fully abstract translations between functional languages*. In: *Proc. of POPL*, ACM, pp. 245–254, doi:10.1145/99583.99617.

[56] D. Sangiorgi (1994): *An investigation into functions as processes*. In: *Proc. of MFPS*, *LNCS* 802, pp. 143–159, doi:10.1007/3-540-58027-1_7.

[57] D. Sangiorgi (2009): *On the Origins of Bisimulation and Coinduction*. ACM Transactions on Programming Languages and Systems (TOPLAS) 31(4), pp. 1–15, doi:10.1145/1516507.1516510.

[58] B. Victor & J. Parrow (1996): *Constraints as Processes*. In: *Proc. of CONCUR*, *LNCS* 1119, pp. 389–405, doi:10.1007/3-540-61604-7_66.

[59] M.G. Vigliotti, I. Phillips & C. Palamidessi (2007): *Tutorial on separation results in process calculi via leader election problems*. Theoretical Computer Science 388(1–3), pp. 267–289, doi:10.1016/j.tcs.2007.09.001.

[60] N. Yoshida (1996): *Graph Types for Monadic Mobile Processes*. In: *Proc. of FST&TCS*, *LNCS* 1180, pp. 371–386, doi:10.1007/3-540-62034-6_64.