The 9th Conference on Mathematics in Engineering: Theory and Applications
Novi Sad, May 31st–June 2nd 2024

PROCESS CALCULI COMPARISONS: A JOURNEY THROUGH ENCODINGS [1]

Jovana Dedeić [2]

https://doi.org/10.24867/META.2024.04 Professional paper

Abstract. This paper provides a comprehensive literature review on the comparison of process calculi, specifically focusing on the encodings. It explores systematic methods for evaluating expressive power through encodings or proofs of their absence. The survey includes commonly used encodability criteria, general frameworks for assessing encoding quality, and methods for comparing these criteria. The insights gained from this review contribute to a better understanding of the concepts of encodings.

AMS Mathematics Subject Classification (2020): 03B30, 68V20

Key words and phrases: concurrent systems, process calculi, encodings, encodability criteria, expressiveness

1. Introduction

The complexity of programs necessitates advanced models for analysis, which highlights the use of formal methods to analyze the properties of complex systems. Formal methods rely on mathematical and logical frameworks for specifying and verifying intricate systems. They utilize modeling languages with precise mathematical syntax and semantics, enabling the demonstration of system properties and verification through mathematical proofs. Examples of formal methods for concurrency include Petri nets, communicating state machines, and process calculi.

In the following, this paper offers an extensive survey of existing literature concerning the evaluation of process calculi, with a specific emphasis on encoding techniques.

2. Expressiveness of Concurrent Calculi

The concept of expressive power in programming languages traces back to the late 1960s, notably with Landin's [12] work on a unified framework for describing language families. Felleisen's [7] framework in the 1990s contributed to studying relative expressiveness, emphasizing eliminable syntactic symbols and definitional extensions between languages. Mitchell [15] and Riecke [25] in 1993

[1]The author acknowledge the financial support of Department of Fundamentals Sciences, Faculty of Technical Sciences, University of Novi Sad, in the frame of Project ”Improving the teaching process in the English language in fundamental disciplines”.

[2]Department of Fundamental Sciences, Faculty of Technical Sciences, University of Novi Sad, e-mail: radenovicj@uns.ac.rs

analyze abstraction-preserving reductions in functional languages, focusing on their impact on program abstraction and expressive power.

Shapiro [27] in 1989 was the first to study expressiveness issues for concurrent languages. He suggested using embedding as a method to compare concurrent logic programming languages that are relatively similar, enabling a focused examination of their distinctions.

The introduction of the π-calculus in the early 1990s significantly advanced the exploration of expressiveness issues within process calculi. This simplicity and adaptability of name-passing, as demonstrated in the π-calculus, spurred numerous works proposing variants or extensions of it. Expressiveness studies for the π-calculus were essential for understanding its fundamental properties, identifying its inherent sources of expressiveness, and exploring relationships between its variants. Examples include studies on polyadic-to-monadic π-calculus translation properties ([24, 30]), point-to-point versus broadcasting communication ([6]), different choice operators ([16, 17]), mechanisms for synchronous and asynchronous communication ([1, 21]), and λ-calculus into the π-calculus [14]. Also, there are comparing various subcalculi of the π-calculus [9], comparing different process calculi and some separation results [2, 3, 4, 11, 13, 22, 23, 26, 28]. Due to varied motivations, each work proposed its own definition of encoding based on specific working intuitions or necessities.

Calculi undergo evaluation, with expressiveness being a key criterion for their assessment. Yet, the theory of concurrency lacks a formal definition of language expressiveness. Given the diversity of concurrency models, a unified theory that encompasses them all is unlikely [10].

In the study of expressiveness, the *concept of encoding* plays a crucial role, as outlined in [**?**]. This encoding, represented as $[\![\cdot]\!]$, translates terms from a *source calculus* to terms in a *target calculus*, adhering to correctness criteria that encompass both *structural* and *semantic* dimensions of the function $[\![\cdot]\!]$. The challenge lies in defining these criteria due to diverse practical needs, which hinders the development of a unified theory for language comparison. Various works in the literature, including those in [8, 19, 20, 21], emphasize the lack of consensus regarding a standardized set of criteria for meaningful encoding. These criteria are often customized to specific analysis requirements. Furthermore, in [20], the author underscores the significance of systematically comparing the increasing number of process models, highlighting a pivotal area of research in the field.

The concept of expressiveness raises questions about its purpose. Expressiveness studies typically focus on two key aspects: *encodability* and *non-encodability.* Encodability investigates the presence of an encoding, while non-encodability deals with the absence of such an encoding. Consider two languages $\mathcal{L}_1$ and $\mathcal{L}_2$. To establish that $\mathcal{L}_1$ is more expressive than $\mathcal{L}_2$, both encodability and non-encodability results must be provided. This means presenting/proving an encoding $[\![\cdot]\!] : \mathcal{L}_2 \longrightarrow \mathcal{L}_1$ and simultaneously demonstrating that an encoding $[\![\cdot]\!] : \mathcal{L}_1 \longrightarrow \mathcal{L}_2$ does not exist.

Another classification in the literature involves *absolute* and *relative expressiveness.* Absolute expressiveness focuses on a single process calculus, leading to either positive or negative absolute results depending on the ability to solve specific problems. Conversely, relative expressiveness compares two languages, determining whether they have the same expressive power or evaluating the impact of specific operators on their expressiveness.

2.1. The Notation of Encoding

As stated in [5], an encoding function maps processes from a source calculus to a target calculus, indicating that the target language is as expressive as the source, or vice versa if no encoding exists. This approach, combining positive and negative encodability results, helps establish differences in expressivity between languages. Translations are often seen as syntax mappings between languages ($\mathcal{L}_\mathbf{s}$ to $\mathcal{L}_\mathbf{t}$), where $\mathcal{P}_\mathbf{s}$ and $\mathcal{P}_\mathbf{t}$ represent sets of process terms. Trivial mappings, like translating every process to inaction, are valid but don't reflect full expressive power. To evaluate encoding quality and avoid trivial mappings, encodings are assessed using a set of correctness criteria. References for further exploration of correctness criteria include [8, 18, 20, 29, 31].

It is common to relate the sources and target calculus through valid encodings (simply encodings). To define valid encodings, we adopt five correctness criteria formulated by Gorla [8]: (1) compositionality, (2) name invariance, (3) operational correspondence, (4) divergence reflection, (5) success sensitiveness. The first two criteria are structural criteria, while the other three are semantic criteria. Structural criteria describe the static structure of the encoding, whereas the semantic criteria describe its dynamics – how the behavior of encoded terms relates to that of source terms, and vice versa. As stated in [20], structural criteria are needed in order to measure the expressiveness of operators in contrast to expressiveness of terms. As for semantic criteria, operational correspondence is divided into *completeness* and *soundness* properties: the former ensures that the behavior of a source process is preserved by the translation in the target calculus; the latter ensures that the behavior of a translated (target) process corresponds to that of some source process. Divergence reflection ensures that a translation does not introduce spurious infinite computations, whereas success sensitiveness requires that source and translated terms behave in the same way with respect to some notion of *success*.

Following [4, 5, 8], we start by defining an abstract notion of calculus:

Definition 2.1 (Calculus [5]). We define a *calculus* as a triple $(\mathcal{P}, \longrightarrow, \approx)$, where: $\mathcal{P}$ is a set of processes; $\longrightarrow$ is its associated reduction semantics, which specifies how a process computes on its own; $\approx$ is an equality on processes, useful to describe the abstract behavior of a process, which is a congruence at least with respect to parallel composition.

We will further assume a countably infinite set of names, usually denoted $\mathcal{N}$. Accordingly, the abstract definition of encoding refers to those names.

Definition 2.2 (Encoding [5]). Let $\mathcal{N}_\mathbf{s}$ and $\mathcal{N}_\mathbf{t}$ be countably infinite sets of source and target names, respectively. An *encoding* of the source calculus $(\mathcal{P}_\mathbf{s}, \longrightarrow_\mathbf{s}, \approx_\mathbf{s})$ into the target calculus $(\mathcal{P}_\mathbf{t}, \longrightarrow_\mathbf{t}, \approx_\mathbf{t})$ is a tuple $(\llbracket\cdot\rrbracket, \varphi_{\llbracket\cdot\rrbracket})$ where $\llbracket\cdot\rrbracket : \mathcal{P}_\mathbf{s} \longrightarrow \mathcal{P}_\mathbf{t}$ denotes a *translation* and $\varphi_{\llbracket\cdot\rrbracket} : \mathcal{N}_\mathbf{s} \longrightarrow \mathcal{N}_\mathbf{t}$ denotes a *renaming policy* for $\llbracket\cdot\rrbracket$.

The renaming policy defines the way names from the source calculus are translated into the target calculus. A valid encoding cannot depend on the particular names involved in source processes.

As in [5], we shall use the following notations. We write $\longrightarrow^*$ to denote the reflexive, transitive closure of $\longrightarrow$. Also, given $k \geq 1$, we will write $P \longrightarrow^k P'$ to denote k consecutive reduction steps leading from P to P'. That is, $P_1 \longrightarrow^k P_{k+1}$ holds whenever there exist $P_2, \dots, P_k$ such that $P_1 \longrightarrow P_2 \longrightarrow \cdots \longrightarrow P_k \longrightarrow P_{k+1}$.

We apply compositionality, as per [4], employing a context that combines translated subterms based on the source operator's combination of subterms.

In this paper, for definition of the operational correspondence we follow more strict criteria than Gorla [8]. We rely on [4, 5] form of operational completeness that explicitly describes the number of steps required to mimic a step in the source language. Also, for divergence reflection we use the following definition:

Definition 2.3 (Divergence [5])**.** A process P diverges, written $P \longrightarrow^\omega$, if there exists an infinite sequence of processes $\{P_i\}_{i\geq 0}$ such that $P = P_0$ and for any i, $P_i \longrightarrow P_{i+1}$.

To formulate success sensitiveness, as in [5], we assume that both source and target calculi contain the same success process $\checkmark$. Also, we assume that $\Downarrow$ is a predicate that asserts reducibility (in a "may" modality) to a process containing an unguarded occurrence of $\checkmark$.

Definition 2.4 (Success [5])**.** Let $(\mathcal{P}, \longrightarrow, \approx)$ be a calculus. A process $P \in \mathcal{P}$ (may)-succeeds, denoted $P \Downarrow$, if it is reducible to a process containing an unguarded occurrence of $\checkmark$, i.e., if $P \longrightarrow^* P'$ and $P' = C[\checkmark]$ for some P' and context $C[\bullet]$.

In the following definition, we formally present the *five criteria* for valid encoding:

Definition 2.5 (Valid Encoding [5])**.** Let $\mathcal{L}_{\mathbf{s}} = (\mathcal{P}_{\mathbf{s}}, \longrightarrow_{\mathbf{s}}, \approx_{\mathbf{s}})$ and $\mathcal{L}_{\mathbf{t}} = (\mathcal{P}_{\mathbf{t}}, \longrightarrow_{\mathbf{t}}, \approx_{\mathbf{t}})$ be source and target calculi, respectively, each with countably infinite sets of names $\mathcal{N}_{\mathbf{s}}$ and $\mathcal{N}_{\mathbf{t}}$. An encoding $(\llbracket\cdot\rrbracket, \varphi_{\llbracket\cdot\rrbracket})$, where $\llbracket\cdot\rrbracket : \mathcal{P}_{\mathbf{s}} \longrightarrow \mathcal{P}_{\mathbf{t}}$ and $\varphi_{\llbracket\cdot\rrbracket} : \mathcal{N}_{\mathbf{s}} \longrightarrow \mathcal{N}_{\mathbf{t}}$, is a *valid encoding* if it satisfies the following criteria:

(1) **Compositionality**: $\llbracket\cdot\rrbracket$ is *compositional* if for every n-ary ($n \geq 1$) operator $\mathtt{op}$ on $\mathcal{P}_s$ and for every set of names $\mathtt{N}$ there is an n-adic context $C^{\mathtt{N}}_{\mathtt{op}}[\bullet_1, \dots, \bullet_n]$ such that, for all $P_1, \dots, P_n$ with $\mathtt{fn}(P_1, \dots, P_n) \subseteq \mathtt{N}$ it holds that $\llbracket \mathtt{op}(P_1, \dots, P_n)\rrbracket = C^{\mathtt{N}}_{\mathtt{op}}[\llbracket P_1\rrbracket, \dots, \llbracket P_n\rrbracket]$.

(2) **Name invariance**: $\llbracket\cdot\rrbracket$ is *name invariant* if for every substitution $\sigma : \mathcal{N}_{\mathbf{s}} \longrightarrow \mathcal{N}_{\mathbf{s}}$ there is a substitution $\sigma' : \mathcal{N}_{\mathbf{t}} \longrightarrow \mathcal{N}_{\mathbf{t}}$ such that (i) for every $a \in \mathcal{N}_{\mathbf{s}} : \varphi_{\llbracket\cdot\rrbracket}(\sigma(a)) = \sigma'(\varphi_{\llbracket\cdot\rrbracket}(a))$ and (ii) $\llbracket\sigma(P)\rrbracket = \sigma'(\llbracket P\rrbracket)$.

(3) **Operational correspondence**: $\llbracket\cdot\rrbracket$ is *operational corresponding* if it satisfies the two requirements:

a) **Completeness**: If $P \longrightarrow_{\mathbf{s}} Q$ then there exists k such that $\llbracket P\rrbracket \longrightarrow^k_{\mathbf{t}} \approx_{\mathbf{t}} \llbracket Q\rrbracket$.

b) **Soundness**: If $\llbracket P\rrbracket \longrightarrow^*_{\mathbf{t}} R$ then there exists P' such that $P \longrightarrow^*_{\mathbf{s}} P'$ and $R \longrightarrow^*_{\mathbf{t}} \approx_{\mathbf{t}} \llbracket P'\rrbracket$.

(4) **Divergence reflection**: $[\![\cdot]\!]$ *reflects divergence* if, for every P such that $[\![P]\!] \longrightarrow_{\mathbf{t}}^{\omega}$, it holds that $P \longrightarrow_{\mathbf{s}}^{\omega}$.

(5) **Success sensitiveness**: $[\![\cdot]\!]$ is *success sensitive* if, for every $P \in \mathcal{P}_{\mathbf{s}}$, it holds that $P \Downarrow$ if and only if $[\![P]\!] \Downarrow$.

Interested readers are encouraged to refer to [4, 5] to see how the authors defined another criterion for their encodings, known as "efficiency". There, the authors consider the number of reduction steps required in the target language to mimic the behavior of the source language.

References

[1] D. Cacciagrano, F. Corrardini, and C. Palamidessi, "Separation of synchronous and asynchronous communication via testing", *Theor. Comput. Sci.* 386, 3, pp. 218–235, 2007.

[2] M. Carbone and S. Maffeis, "On the expressive power of polyadic synchronisation in pi-calculus", *Nordic Journal of Computing*, 10, 2, pp. 70–98, 2003.

[3] O. Dardha, E. Giachino, and D. Sangiorgi, "Session types revisited", *Information and Computation*, 256, pp. 253–286, 2017.

[4] J. Dedeić, J. Pantović, and J. A. Pérez, "On primitives for compensation handling as adaptable processes", *Journal of Logical and Algebraic Methods in Programming*, p. 100675, 2021.

[5] J. Dedeić, *Relative Expressiveness of Process Calculi with Dynamic Update and Runtime Adaptation*, PhD dissertation, Faculty of Technical Sciences, University of Novi Sad, 2022.

[6] C. Ene and T. Muntean, "Expressiveness of point-to-point versus broadcast communications", *In Proc. of FCT. Lecture Notes in Computer Science*, vol. 1684, Springer, pp. 258–268, 1999.

[7] M. Felleisen, "On the expressive power of programming languages", *Sci. Comput. Program.*, 17, 1-3, pp. 35–75, 1991.

[8] D. Gorla, "Towards a unified approach to encodability and separation results for process calculi", *Inf. Comput.*, 208, 9, pp. 1031–1053, 2010.

[9] K. Honda and M. Tokoro, "An object calculus for asynchronous communication", *In Proc. of ECOOP'91, Geneva, Switzerland, July 15-19, 1991, Proceedings*, vol. 512 of Lecture Notes in Computer Science, pp. 133–147. Springer, 1991.

[10] J. A. Pérez, *Higher-Order Concurrency: Expressiveness and Decidability Results*, PhD dissertation,, University of Bologna, Department of Computer Science, 2010.

[11] I. Lanese, J. A. Pérez, D. Sangiorgi, and A. Schmitt, "On the expressiveness and decidability of higher-order process calculi", *Information and Computation*, 209, 2, pp. 198–226, 2011.

[12] P. J. Landin, "The Next 700 Programming Languages", *Communications of the ACM*, 9, 3, pp. 157–166, 1966.

[13] D. Medić and C. A. Mezzina, "Static VS dynamic reversibility in CCS", *8th International Conference, RC 2016, Bologna, Italy, July 7-8, 2016, Proceedings*, vol. 9720 of Lecture Notes in Computer Science, pp. 36–51. Springer, 2016.

[14] R. Milner, "Functions as processes", *Mathematical Structures in Computer Science*, 2, 2, pp. 119–141, 1992.

[15] J. C. Mittchell, "On abstraction and the expressive power of programming languages", *Sci. Comput. Program.*, 21, 2, pp. 141–163, 1993.

[16] U. Nestmann, "What is a "good" encoding of guarded choice?", *Inf. Comput.*, 156, 1-2, pp. 287–319, 2000.

[17] U. Nestmann and B. C. Pierce, "Decoding choice encodings", *Inf. Comput.*, 163, 1, pp. 1–59, 2000.

[18] U. Nestmann, *On Determinacy and Nondeterminacy in Concurrent Programming*, PhD dissertation, Universität Erlangen-Nürnberg, 1996.

[19] U. Nestmann, "Welcome to the jungle: A subjective guide to mobile process calculi", *In Proceedings of the 17th International Conference on Concurrency Theory*, CONCUR'06, pp. 52–63, Berlin, Heidelberg, 2006.

[20] J. Parrow, "Expressiveness of process algebras", *Electronic Notes in Theoretical Computer Science*, 209, pp. 173–186, 2008.

[21] C. Palamidessi, "Comparing the expressive power of the synchronous and asynchronous pi-calculi", *Mathematical Structures in Computer Science*, 13, 5, pp. 685–719, 2003.

[22] C. Palamidessi, V. A. Saraswat, F. D. Valencia, and B. Victor, "On the expressiveness of linearity vs persistence in the asynchronous π-calculus", *In Proc. of the 21st IEEE Symposium on Logic in Computer Science (LICS 2006)*, Seattle, WA, USA, August 12-15, 2006, pp. 59–68. IEEE Computer Society, 2006.

[23] I. Prokic and H. T. Vieira, "The C_π-calculus: A model for confidential name passing", *Journal of Logical and Algebraic Methods in Programming*, 119, p. 100622, 2021.

[24] P. Quaglia and D. Walker, "Types and full abstraction for polyadic pi-calculus", *Inf. Comput.*, 200, 2, pp. 215–246, 2005.

[25] J. G. Riecke, "Fully abstract translations between functional languages", *Mathematical Structures in Computer Science*, 3, 4, pp. 387–415, 1993.

[26] D. Sangiorgi, *Expressing Mobility in Process Algebras: First-Order and Higher-Order Paradigms*, PhD thesis, University of Edinburgh, UK, 1993.

[27] E. Y. Shapiro, "The family of concurrent logic programming languages", *ACM Comput. Surv.*, 21, 3, pp. 413–510, 1989.

[28] M. G. Vigliotti, I. Phillips, and C. Palamidessi, "Tutorial on separation results in process calculi via leader election problems", *Theoretical Computer Science*, 388, 1, pp. 267–289, 2007.

[29] N. Yoshida, "Minimality and separation results on asynchronous mobile processes - representability theorems by concurrent combinators", *Theoretical Computer Science*, 274, 1, pp. 231–276, 2002.

[30] N. Yoshida, "Graph types for monadic mobile processes", *In Proc. of FSTTCS. Lecture Notes in Computer Science*, vol. 1180, Springer, pp. 371–386, 1996.

[31] K. Peters, *Translational expressiveness: comparing process calculi using encodings*, PhD dissertation, Technische Universität Berlin, Fakultät IV - Elektrotechnik und Informatik, Berlin, 2012.