Intersection Types Meet Session Types

Joseph Paulus
University of Oxford, UK

Daniele Nantes-Sobrinho
Imperial College London, UK

Jorge A. Pérez
University of Groningen, The Netherlands

Abstract Milner's seminal work on encodings of the λ-calculus into the π-calculus ("functions as processes" [20]) explains how *interaction* in π subsumes *evaluation* in λ. It opened a research strand on formal connections between sequential and concurrent calculi, covering untyped and typed regimes (see, e.g., [26, 4, 2, 27, 16, 28]). In a series of recent works ([23, 24, 25, 12, 13]) we have extended this line of work by considering calculi in which computation is *non-deterministic* and may be subject to *failures*—two relevant features in sequential and concurrent programming models. Because we consider *typed* source and target languages, an interesting byproduct of these expressivity results is a new connection between (non-idempotent) *intersection types* (in λ) with *session types* (in π). In this note, we briefly review this line of work, and outline directions for future developments.

Setting We focus on *typed* calculi and study how non-determinism and failures interact with *resource-aware* computation. In sequential calculi, *non-idempotent intersection types* offer one fruitful perspective at resource-awareness (see, e.g., [8, 17, 18, 21, 5]). Because non-idempotency amounts to distinguish between types σ and $\sigma \wedge \sigma$, this class of intersection types can "count" different resources and enforce quantitative guarantees. In concurrent calculi, resource-awareness has been much studied using *linear types*. Linearity ensures that process actions occur exactly once, which is key to enforce protocol correctness. In particular, *session types* [14, 15] specify the protocols that channels must respect; this typing discipline exploits linearity to ensure absence of communication errors and stuck processes. To our knowledge, we are the first to consider connections between calculi adopting these two distinct views of resource-awareness; we do so by relating typed models of sequential and concurrent computation.

On the sequential side, we introduce $\lambda^{\lightning}_{\oplus}$: a λ-calculus with resources, non-determinism, and failures, which distills key elements from λ-calculi studied in [3, 22]. Evaluation in $\lambda^{\lightning}_{\oplus}$ considers *bags* of resources, and determines alternative executions governed by non-determinism. Failure results from a lack or excess of resources (terms), and is captured by the term $\mathtt{fail}^{\widetilde{x}}$, where $\widetilde{x}$ denotes a sequence of variables. Non-determinism in $\lambda^{\lightning}_{\oplus}$ is *confluent*: intuitively, given M and N with reductions $M \longrightarrow M'$ and $N \longrightarrow N'$, the non-deterministic sum $M \oplus N$ reduces to $M' \oplus N'$. (In contrast, under a *non-confluent* approach, as in, e.g., [7], the non-deterministic sum $M \oplus N$ reduces to either M or N.)

On the concurrent side, we consider $\mathsf{s}\pi$: a session-typed π-calculus with non-determinism and failure, proposed in [6]. $\mathsf{s}\pi$ rests upon a Curry-Howard correspondence between session types and (classical) linear logic, extended with modalities that express *non-deterministic protocols* that may succeed or fail. In $\mathsf{s}\pi$, non-determinism is confluent.

Contributions Our paper [23] (later extended into [25]) presents the first formal connection between a λ-calculus with non-idempotent intersection types and a π-calculus with session types. Specifically, in that work we develop the following contributions:

1. **The resource calculus** $\lambda^{\lightning}_{\oplus}$, a new calculus that distills the distinctive elements from previous resource calculi [4, 22], while offering an explicit treatment of failures in a setting with confluent

Submitted to:
ITRS 2024

non-determinism.

We develop the syntax, semantics, and essential meta-theoretical results for $\lambda^{\lightning}_{\oplus}$. In particular, using non-idempotent intersection types, we define *well-typed* (fail-free) expressions and *well-formed* (fail-prone) expressions in $\lambda^{\lightning}_{\oplus}$ and establish their properties.

2. **An encoding of $\lambda^{\lightning}_{\oplus}$ into $\mathsf{s}\pi$**, proven correct following established criteria in the realm of relative expressiveness for concurrency, which attest to an encoding's quality [10, 19]. We consider *operational correspondence* (including *completeness* and *soundness*), *success sensitiveness*, and *compositionality*.
3. **An encoding of intersection types into session types**. Our encoding also enjoys *type preservation*: in addition to the encoding of terms into processes, we also have an encoding of intersection types for $\lambda^{\lightning}_{\oplus}$ into session types for $\mathsf{s}\pi$. Such an encoding is interesting, because it precisely describes how typed interaction protocols (given by session types) can codify sequential evaluation in which absence and excess of resources leads to failures (as governed by intersection types).

Extensions We have considered extensions of our approach and results in [23, 25], in two directions.

- **Unrestricted Resources.** While in $\lambda^{\lightning}_{\oplus}$ resources in bags are strictly linear (i.e., usable exactly once), the case of *unrestricted* (non-linear) resources is also relevant. In [24], we extend our languages and results accordingly: we develop $\lambda^{!\lightning}_{\oplus}$, an extension of $\lambda^{\lightning}_{\oplus}$ in which bags contain resources that can be consumed an arbitrary number of times (possibly zero). This extension requires a careful treatment of resource fetching, as a variable can be substituted with either a linear resource or an unrestricted one, but not both. Using $\lambda^{!\lightning}_{\oplus}$, the two contributions above (the calculus itself and its correct encoding into session-typed processes) are lifted to the setting in which linear and unrestricted resources coexist.
- **Non-Confluent Non-Determinism**. Although results involving confluent non-determinism are significant, usual (non-confluent) non-determinism remains of undiscussed convenience in formal modeling. In [12, 13], we adapt our developments in [23, 24, 25] (which rely on confluent non-determinism), to the case of non-confluent non-determinism. We define new functional and concurrent calculi, following the approach and typing disciplines used for the confluent case. As in our previous works, the translation of types abstractly describes how non-deterministic fetches are codified as non-deterministic session protocols. Interestingly, the translation of types in [12, 13] is the same as in [24]. While not extremely surprising, it is pleasant that the translation of types remains unchanged across different translations with our confluent and non-confluent non-determinism.

The Proposed Talk The proposed talk at ITRS'24 will detail how these contributions entail different challenges. The first is bridging the different mechanisms for resource-awareness involved (i.e., intersection types in $\lambda^{\lightning}_{\oplus}$, session types in $\mathsf{s}\pi$). A direct encoding of $\lambda^{\lightning}_{\oplus}$ into $\mathsf{s}\pi$ is far from obvious, as multiple occurrences of a variable in $\lambda^{\lightning}_{\oplus}$ must be accommodated into the linear setting of $\mathsf{s}\pi$. To overcome this challenge, we introduce a variant of $\lambda^{\lightning}_{\oplus}$ with *sharing* [11, 9], dubbed $\widehat{\lambda}^{\lightning}_{\oplus}$. Our encoding of $\lambda^{\lightning}_{\oplus}$ expressions into $\mathsf{s}\pi$ processes is then in two steps. We first define a correct encoding from $\lambda^{\lightning}_{\oplus}$ to $\widehat{\lambda}^{\lightning}_{\oplus}$, which conveniently "atomizes" occurrences of the same variable. Then, we define another correct encoding, from $\widehat{\lambda}^{\lightning}_{\oplus}$ to $\mathsf{s}\pi$, which extends Milner's with constructs for non-determinism.

Another challenge is framing failures in $\lambda_{\oplus}^{\lightning}$ (undesirable computations) as well-typed $\mathtt{s}\pi$ processes. Using intersection types, we define *well-formed* $\lambda_{\oplus}^{\lightning}$ expressions, which can fail, in two stages. First, we consider $\lambda_{\oplus}$, the sub-language of $\lambda_{\oplus}^{\lightning}$ without $\mathtt{fail}^{\widetilde{x}}$. We give an intersection type system for $\lambda_{\oplus}$ to regulate fail-free evaluation. Well-formed expressions are then defined on top of well-typed $\lambda_{\oplus}$ expressions. We show that $\mathtt{s}\pi$ can correctly encode the fail-free $\lambda_{\oplus}$ but, more interestingly, also well-formed $\lambda_{\oplus}^{\lightning}$ expressions, which are fail-prone.

As we have seen, up to now the work on $\lambda_{\oplus}^{\lightning}$ (and its several variants) has focused on their role as representative source languages for a comparison of typed models of computation with non-determinism, across sequential and concurrent scenarios. In *future work*, we intend to deepen on the role and properties of $\lambda_{\oplus}^{\lightning}$ as a typed programming language, in particular exploiting the *quantitative* perspective enabled by non-idempotent session types (i.e., extracting measures/bounds on computational steps, see, e.g., [1]). This study appears as an interesting (and necessary) step for the development of concurrent extensions of $\lambda_{\oplus}^{\lightning}$ with message-passing concurrency governed by sessions, which are immediately enabled by our correct translations into $\mathtt{s}\pi$.

References

[1] Beniamino Accattoli, Stéphane Graham-Lengrand & Delia Kesner (2020): *Tight typings and split bounds, fully developed*. *J. Funct. Program.* 30, p. e14, doi:10.1017/S095679682000012X. Available at `https://doi.org/10.1017/S095679682000012X`.

[2] Martin Berger, Kohei Honda & Nobuko Yoshida (2003): *Genericity and the pi-Calculus*. In Andrew D. Gordon, editor: *Foundations of Software Science and Computational Structures, 6th International Conference, FOSSACS 2003 Held as Part of the Joint European Conference on Theory and Practice of Software, ETAPS 2003, Warsaw, Poland, April 7-11, 2003, Proceedings*, *Lecture Notes in Computer Science* 2620, Springer, pp. 103–119, doi:10.1007/3-540-36576-1_7. Available at `https://doi.org/10.1007/3-540-36576-1_7`.

[3] Gérard Boudol (1993): *The Lambda-Calculus with Multiplicities (Abstract)*. In Eike Best, editor: *CONCUR '93, Hildesheim, Germany, August 23-26, 1993, Proceedings*, *Lecture Notes in Computer Science* 715, Springer, pp. 1–6, doi:10.1007/3-540-57208-2_1. Available at `https://doi.org/10.1007/3-540-57208-2_1`.

[4] Gérard Boudol & Cosimo Laneve (2000): *lambda-calculus, multiplicities, and the pi-calculus*. In: *Proof, Language, and Interaction, Essays in Honour of Robin Milner*, pp. 659–690.

[5] Antonio Bucciarelli, Delia Kesner & Daniel Ventura (2017): *Non-idempotent intersection types for the Lambda-Calculus*. *Logic Journal of the IGPL* 25(4), pp. 431–464.

[6] Luís Caires & Jorge A. Pérez (2017): *Linearity, Control Effects, and Behavioral Types*. In Hongseok Yang, editor: *Programming Languages and Systems - 26th European Symposium on Programming, ESOP 2017, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2017, Uppsala, Sweden, April 22-29, 2017, Proceedings*, *Lecture Notes in Computer Science* 10201, Springer, pp. 229–259, doi:10.1007/978-3-662-54434-1_9. Available at `https://doi.org/10.1007/978-3-662-54434-1_9`.

[7] Mariangiola Dezani-Ciancaglini, Ugo de'Liguoro & Adolfo Piperno (1993): *Filter Models for a Parallel and Non Deterministic Lambda-Calculus*. In Andrzej M. Borzyszkowski & Stefan Sokolowski, editors: *Mathematical Foundations of Computer Science 1993, 18th International Symposium, MFCS'93, Gdansk, Poland, August 30 - September 3, 1993, Proceedings*, *Lecture Notes in Computer Science* 711, Springer, pp. 403–412, doi:10.1007/3-540-57182-5_32. Available at `https://doi.org/10.1007/3-540-57182-5_32`.

[8] Philippa Gardner (1994): *Discovering Needed Reductions Using Type Theory*. In Masami Hagiya & John C. Mitchell, editors: *Theoretical Aspects of Computer Software, International Conference TACS '94, Sendai,*

Japan, April 19-22, 1994, Proceedings, *Lecture Notes in Computer Science* 789, Springer, pp. 555–574, doi:10.1007/3-540-57887-0_115. Available at https://doi.org/10.1007/3-540-57887-0_115.

[9] Silvia Ghilezan, Jelena Ivetic, Pierre Lescanne & Silvia Likavec (2011): *Intersection Types for the Resource Control Lambda Calculi*. In: *Theoretical Aspects of Computing - ICTAC 2011 - 8th International Colloquium, Johannesburg, South Africa, August 31 - September 2, 2011. Proceedings*, pp. 116–134, doi:10.1007/978-3-642-23283-1_10. Available at https://doi.org/10.1007/978-3-642-23283-1_10.

[10] Daniele Gorla (2010): *Towards a unified approach to encodability and separation results for process calculi*. *Inf. Comput.* 208(9), pp. 1031–1053, doi:10.1016/j.ic.2010.05.002. Available at https://doi.org/10.1016/j.ic.2010.05.002.

[11] Tom Gundersen, Willem Heijltjes & Michel Parigot (2013): *Atomic Lambda Calculus: A Typed Lambda-Calculus with Explicit Sharing*. In: *28th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2013, New Orleans, LA, USA, June 25-28, 2013*, pp. 311–320, doi:10.1109/LICS.2013.37. Available at https://doi.org/10.1109/LICS.2013.37.

[12] Bas van den Heuvel, Joseph W. N. Paulus, Daniele Nantes-Sobrinho & Jorge A. Pérez (2023): *Typed Nondeterminism in Functional and Concurrent Calculi*. In Chung-Kil Hur, editor: *Programming Languages and Systems - 21st Asian Symposium, APLAS 2023, Taipei, Taiwan, November 26-29, 2023, Proceedings*, *Lecture Notes in Computer Science* 14405, Springer, pp. 112–132, doi:10.1007/978-981-99-8311-7_6. Available at https://doi.org/10.1007/978-981-99-8311-7_6.

[13] Bas van den Heuvel, Joseph W. N. Paulus, Daniele Nantes-Sobrinho & Jorge A. Pérez (2024): *Typed Nondeterminism in Concurrent Calculi: The Eager Way*. In: *Proceedings of the 40th Conference on the Mathematical Foundations of Programming Semantics, MFPS XL*, EPTICS, EpiSciences. To appear.

[14] Kohei Honda (1993): *Types for Dyadic Interaction*. In Eike Best, editor: *CONCUR '93, Hildesheim, Germany, August 23-26, 1993, Proceedings*, *Lecture Notes in Computer Science* 715, Springer, pp. 509–523, doi:10.1007/3-540-57208-2_35. Available at https://doi.org/10.1007/3-540-57208-2_35.

[15] Kohei Honda, Vasco Thudichum Vasconcelos & Makoto Kubo (1998): *Language Primitives and Type Discipline for Structured Communication-Based Programming*. In Chris Hankin, editor: *Programming Languages and Systems - ESOP'98, 7th European Symposium on Programming, Held as Part of the European Joint Conferences on the Theory and Practice of Software, ETAPS'98, Lisbon, Portugal, March 28 - April 4, 1998, Proceedings*, *Lecture Notes in Computer Science* 1381, Springer, pp. 122–138, doi:10.1007/BFb0053567. Available at https://doi.org/10.1007/BFb0053567.

[16] Kohei Honda, Nobuko Yoshida & Martin Berger (2014): *Process Types as a Descriptive Tool for Interaction - Control and the Pi-Calculus*. In Gilles Dowek, editor: *Rewriting and Typed Lambda Calculi - Joint International Conference, RTA-TLCA 2014, Held as Part of the Vienna Summer of Logic, VSL 2014, Vienna, Austria, July 14-17, 2014. Proceedings*, *Lecture Notes in Computer Science* 8560, Springer, pp. 1–20, doi:10.1007/978-3-319-08918-8_1. Available at https://doi.org/10.1007/978-3-319-08918-8_1.

[17] A. J. Kfoury (2000): *A linearization of the Lambda-calculus and consequences*. *J. Log. Comput.* 10(3), pp. 411–436, doi:10.1093/logcom/10.3.411. Available at https://doi.org/10.1093/logcom/10.3.411.

[18] A. J. Kfoury & J. B. Wells (2004): *Principality and type inference for intersection types using expansion variables*. *Theor. Comput. Sci.* 311(1-3), pp. 1–70, doi:10.1016/j.tcs.2003.10.032. Available at https://doi.org/10.1016/j.tcs.2003.10.032.

[19] Dimitrios Kouzapas, Jorge A. Pérez & Nobuko Yoshida (2019): *On the relative expressiveness of higher-order session processes*. *Inf. Comput.* 268, doi:10.1016/j.ic.2019.06.002. Available at https://doi.org/10.1016/j.ic.2019.06.002.

[20] Robin Milner (1992): *Functions as Processes*. *Mathematical Structures in Computer Science* 2(2), pp. 119–141, doi:10.1017/S0960129500001407. Available at https://doi.org/10.1017/S0960129500001407.

[21] Peter Møller Neergaard & Harry G. Mairson (2004): *Types, potency, and idempotency: why nonlinearity and amnesia make a type system work*. In Chris Okasaki & Kathleen Fisher, editors: *Proceedings of the*

Ninth ACM SIGPLAN International Conference on Functional Programming, ICFP 2004, Snow Bird, UT, USA, September 19-21, 2004, ACM, pp. 138–149, doi:10.1145/1016850.1016871. Available at `https://doi.org/10.1145/1016850.1016871`.

[22] Michele Pagani & Simona Ronchi Della Rocca (2010): *Solvability in Resource Lambda-Calculus*. In C.-H. Luke Ong, editor: *Foundations of Software Science and Computational Structures, 13th International Conference, FOSSACS 2010, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2010, Paphos, Cyprus, March 20-28, 2010. Proceedings*, *Lecture Notes in Computer Science* 6014, Springer, pp. 358–373, doi:10.1007/978-3-642-12032-9_25. Available at `https://doi.org/10.1007/978-3-642-12032-9_25`.

[23] Joseph W. N. Paulus, Daniele Nantes-Sobrinho & Jorge A. Pérez (2021): *Non-Deterministic Functions as Non-Deterministic Processes*. In Naoki Kobayashi, editor: *6th International Conference on Formal Structures for Computation and Deduction, FSCD 2021, July 17-24, 2021, Buenos Aires, Argentina (Virtual Conference)*, *LIPIcs* 195, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, pp. 21:1–21:22, doi:10.4230/LIPIcs.FSCD.2021.21. Available at `https://doi.org/10.4230/LIPIcs.FSCD.2021.21`.

[24] Joseph W. N. Paulus, Daniele Nantes-Sobrinho & Jorge A. Pérez (2021): *Types and Terms Translated: Unrestricted Resources in Encoding Functions as Processes*. In Henning Basold, Jesper Cockx & Silvia Ghilezan, editors: *27th International Conference on Types for Proofs and Programs, TYPES 2021, June 14-18, 2021, Leiden, The Netherlands (Virtual Conference)*, *LIPIcs* 239, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, pp. 11:1–11:24, doi:10.4230/LIPIcs.TYPES.2021.11. Available at `https://doi.org/10.4230/LIPIcs.TYPES.2021.11`.

[25] Joseph W. N. Paulus, Daniele Nantes-Sobrinho & Jorge A. Pérez (2023): *Non-Deterministic Functions as Non-Deterministic Processes (Extended Version)*. *Log. Methods Comput. Sci.* 19(4), doi:10.46298/LMCS-19(4:1)2023. Available at `https://doi.org/10.46298/lmcs-19(4:1)2023`.

[26] Davide Sangiorgi (1999): *From lambda to pi; or, Rediscovering continuations*. *Math. Struct. Comput. Sci.* 9(4), pp. 367–401. Available at `http://journals.cambridge.org/action/displayAbstract?aid=44843`.

[27] Bernardo Toninho, Luís Caires & Frank Pfenning (2012): *Functions as Session-Typed Processes*. In Lars Birkedal, editor: *Foundations of Software Science and Computational Structures - 15th International Conference, FOSSACS 2012, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2012, Tallinn, Estonia, March 24 - April 1, 2012. Proceedings*, *Lecture Notes in Computer Science* 7213, Springer, pp. 346–360, doi:10.1007/978-3-642-28729-9_23. Available at `https://doi.org/10.1007/978-3-642-28729-9_23`.

[28] Bernardo Toninho & Nobuko Yoshida (2018): *On Polymorphic Sessions and Functions - A Tale of Two (Fully Abstract) Encodings*. In Amal Ahmed, editor: *27th European Symposium on Programming, ESOP 2018, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2018, Thessaloniki, Greece, April 14-20, 2018, Proceedings*, *Lecture Notes in Computer Science* 10801, Springer, pp. 827–855, doi:10.1007/978-3-319-89884-1_29. Available at `https://doi.org/10.1007/978-3-319-89884-1_29`.