Non-Deterministic Functions as Non-Deterministic Processes

Paulus, Joseph W.N.; Nantes-Sobrinho, Daniele; Pérez, Jorge A.

Published in:
6th International Conference on Formal Structures for Computation and Deduction (FSCD 2021)

DOI:
10.4230/LIPIcs.FSCD.2021.21

Publication date:
2021

Non-Deterministic Functions as Non-Deterministic Processes

Joseph W. N. Paulus
University of Groningen, The Netherlands

Daniele Nantes-Sobrinho
University of Brasília, Brazil

Jorge A. Pérez
University of Groningen, The Netherlands
CWI, Amsterdam, The Netherlands

Abstract

We study encodings of the λ-calculus into the π-calculus in the unexplored case of calculi with *non-determinism* and *failures*. On the sequential side, we consider $\lambda^{\natural}_{\oplus}$, a new non-deterministic calculus in which intersection types control resources (terms); on the concurrent side, we consider $\mathsf{s}\pi$, a π-calculus in which non-determinism and failure rest upon a Curry-Howard correspondence between linear logic and session types. We present a typed encoding of $\lambda^{\natural}_{\oplus}$ into $\mathsf{s}\pi$ and establish its correctness. Our encoding precisely explains the interplay of non-deterministic and fail-prone evaluation in $\lambda^{\natural}_{\oplus}$ via typed processes in $\mathsf{s}\pi$. In particular, it shows how failures in sequential evaluation (absence/excess of resources) can be neatly codified as interaction protocols.

2012 ACM Subject Classification Theory of computation → Type structures; Theory of computation → Process calculi

Keywords and phrases Resource calculi, π-calculus, intersection types, session types, linear logic

Digital Object Identifier 10.4230/LIPIcs.FSCD.2021.21

Related Version Online appendix with omitted proofs and further examples:
Full Version: `https://arxiv.org/abs/2104.14759` [22]

Funding Paulus and Pérez have been partially supported by the Dutch Research Council (NWO) under project No. 016.Vidi.189.046 (Unifying Correctness for Communicating Software).

Acknowledgements We are grateful to the anonymous reviewers for their careful reading and constructive remarks.

1 Introduction

Milner's seminal work on encodings of the λ-calculus into the π-calculus [18] 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., [23, 4, 1, 25, 16, 26]). This paper extends this line of work by tackling a hitherto unexplored angle, namely encodability of calculi in which computation is *non-deterministic* and may be subject to *failures* – two relevant features in sequential and concurrent programming models.

We focus on *typed* calculi and study how non-determinism and failures interact with *resource-aware* computation. In sequential calculi, *non-idempotent intersection types* [2] offer one fruitful perspective at resource-awareness. Because non-idempotency distinguishes 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. To our knowledge, connections between calculi adopting these two distinct views of resource-awareness via types are still to be established. We aim to develop such connections by relating models of sequential and concurrent computation.

On the sequential side, we introduce $\lambda^{\not\lightning}_{\oplus}$: a λ-calculus with resources, non-determinism, and failures, which distills key elements from λ-calculi studied in [3, 21] (§ 2). Evaluation in $\lambda^{\not\lightning}_{\oplus}$ 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 $\mathtt{fail}^{\widetilde{x}}$ (for some variables $\widetilde{x}$). Non-determinism is *non-collapsing*: given M and N with reductions $M \longrightarrow M'$ and $N \longrightarrow N'$, the non-deterministic sum $M + N$ reduces to $M' + N'$. (Under a *collapsing* view, as in, e.g., [8], $M + N$ reduces to either M or N.)

On the concurrent side, we consider $\mathsf{s}\pi$: a session π-calculus with (non-collapsing) non-determinism and failure proposed in [6] (§ 3). Processes in $\mathsf{s}\pi$ are disciplined by *session types* that specify the protocols that the channels of a process must respect. Exploiting linearity, session types ensure absence of communication errors and stuck processes; $\mathsf{s}\pi$ rests upon a Curry-Howard correspondence between session types and (classical) linear logic extended with two modalities that express *non-deterministic protocols* that may succeed or fail.

Contributions. This paper presents the following contributions:

1. **The resource calculus $\boldsymbol{\lambda^{\not\lightning}_{\oplus}}$**, a new calculus that distills the distinguishing elements from previous resource calculi [4, 21], while offering an explicit treatment of failures in a setting with non-collapsing non-determinism. Using intersection types, we define well-typed (fail-free) expressions and well-formed (fail-prone) expressions in $\lambda^{\not\lightning}_{\oplus}$ (see below).
2. **An encoding of $\boldsymbol{\lambda^{\not\lightning}_{\oplus}}$ into $\boldsymbol{\mathsf{s}\pi}$**, proven correct following established criteria [11, 17] (§ 4). These criteria attest to an encoding's quality; we consider *type preservation*, *operational correspondence*, *success sensitiveness*, and *compositionality*. Thanks to these correctness properties, our encoding precisely describes how typed interaction protocols can codify sequential evaluation in which the absence and excess of resources may lead to failures.

These contributions entail different challenges. The first is bridging the different mechanisms for resource-awareness involved (intersection types in $\lambda^{\not\lightning}_{\oplus}$, session types in $\mathsf{s}\pi$). A direct encoding of $\lambda^{\not\lightning}_{\oplus}$ into $\mathsf{s}\pi$ is far from obvious, as multiple occurrences of a variable in $\lambda^{\not\lightning}_{\oplus}$ must be accommodated into the linear setting of $\mathsf{s}\pi$. To overcome this, we introduce $\widehat{\lambda}^{\not\lightning}_{\oplus}$: a variant of $\lambda^{\not\lightning}_{\oplus}$ with *sharing* [13, 10]. This way, we "atomize" occurrences of the same variable, thus simplifying the task of encoding $\lambda^{\not\lightning}_{\oplus}$ expressions into $\mathsf{s}\pi$ processes.

Another challenge is framing failures (undesirable computations) in $\lambda^{\not\lightning}_{\oplus}$ as well-typed $\mathsf{s}\pi$ processes. We define *well-formed* $\lambda^{\not\lightning}_{\oplus}$ expressions, which can lead to failure, in two stages. First, we consider $\lambda_{\oplus}$, the sub-language of $\lambda^{\not\lightning}_{\oplus}$ without $\mathtt{fail}^{\widetilde{x}}$. We give an intersection type system for $\lambda_{\oplus}$ to regulate fail-free evaluation. Well-formed expressions are defined on top of well-typed $\lambda_{\oplus}$ expressions. We show that $\mathsf{s}\pi$ can correctly encode the fail-free $\lambda_{\oplus}$ but, much more interestingly, also well-formed $\lambda^{\not\lightning}_{\oplus}$ expressions, which are fail-prone by definition.

Discussion about our approach and results, and comparisons with related works is in § 5.

2 $\lambda^{\not\lightning}_{\oplus}$: A λ-calculus with Non-Determinism and Failure

The syntax of $\lambda^{\not\lightning}_{\oplus}$ combines elements from calculi studied by Boudol and Laneve [4] and by Pagani and Ronchi della Rocca [21]. We use $x, y, \ldots$ to range over the set of *variables*. We write $\widetilde{x}$ to denote the sequence of pairwise distinct variables $x_1, \ldots, x_k$, for some $k \geq 0$. We write $|\widetilde{x}|$ to denote the length of $\widetilde{x}$.

▶ **Definition 1** (Syntax of $\lambda^{\natural}_{\oplus}$). *The $\lambda^{\natural}_{\oplus}$ calculus is defined by the following grammar:*

$$\begin{array}{lllll} \textit{(Terms)} & M, N, L & ::= & x \mid \lambda x.M \mid (M\ B) \mid M\langle\!\langle {}^{B}/_{x} \rangle\!\rangle \mid \mathtt{fail}^{\widetilde{x}} \\ \textit{(Bags)} & A, B & ::= & \mathtt{1} \mid \wr M \wr \mid A \cdot B \\ \textit{(Expressions)} & \mathbb{M}, \mathbb{N}, \mathbb{L} & ::= & M \mid \mathbb{M} + \mathbb{N} \end{array}$$

We have three syntactic categories: *terms* (in functional position); *bags* (in argument position), which denote multisets of resources; and *expressions*, which are finite formal sums that represent possible results of a computation. Terms are unary expressions: they can be variables, abstractions, and applications. Following [3, 4], the *explicit substitution* of a bag B for a variable x, written $\langle\!\langle {}^{B}/_{x} \rangle\!\rangle$, is also a term. The term $\mathtt{fail}^{\widetilde{x}}$ results from a reduction in which there is a lack or excess of resources to be substituted, where $\widetilde{x}$ denotes a multiset of free variables that are encapsulated within failure.

The empty bag is denoted $\mathtt{1}$. The bag enclosing the term M is $\wr M \wr$. The concatenation of bags B_1 and B_2 is $B_1 \cdot B_2$; this is a commutative and associative operation, where $\mathtt{1}$ is the identity. We treat expressions as *sums*, and use notations such as $\sum_i^n N_i$ for them. Sums are associative and commutative; reordering of the terms in a sum is performed silently.

▶ **Notation 2** (Expressions). *Notation $N \in \mathbb{M}$ denotes that N is part of the sum denoted by $\mathbb{M}$. Similarly, we write $N_i \in B$ to denote that N_i occurs in the bag B, and $B \setminus N_i$ to denote the bag that is obtained by removing one occurrence of the term N_i from B.*

Full details on the reduction semantics and typing system for $\lambda^{\natural}_{\oplus}$ can be found in the appendix and [22].

A Resource Calculus With Sharing

We define a variant of $\lambda^{\natural}_{\oplus}$ with sharing variables, dubbed $\widehat{\lambda}^{\natural}_{\oplus}$, inspired by the work by Gundersen et al. [13] and Ghilezan et al. [10]. In § 4 we shall use $\widehat{\lambda}^{\natural}_{\oplus}$ as intermediate language in our encoding of $\lambda^{\natural}_{\oplus}$ into $\mathsf{s}\pi$.

The syntax of $\widehat{\lambda}^{\natural}_{\oplus}$ only modifies the syntax of $\lambda^{\natural}_{\oplus}$-terms, which is defined by the grammar below; the syntax of bags B and expressions $\mathbb{M}$ is as in Def. 1.

$$\begin{array}{lll} \text{(Terms)} & M, N, L ::= & x \mid \lambda x.(M[\widetilde{x} \leftarrow x]) \mid (M\ B) \mid M\langle\!|N/x|\!\rangle \mid \mathtt{fail}^{\widetilde{x}} \\ & & \mid M[\widetilde{x} \leftarrow x] \mid (M[\widetilde{x} \leftarrow x])\langle\!\langle {}^{B}/_{x} \rangle\!\rangle \end{array}$$

We consider the *sharing construct* $M[\widetilde{x} \leftarrow x]$ and the *explicit linear substitution* $M\langle\!|N/x|\!\rangle$. The term $M[\widetilde{x} \leftarrow x]$ defines the sharing of variables $\widetilde{x}$ occurring in M using x. We shall refer to x as *sharing variable* and to $\widetilde{x}$ as *shared variables*. A variable is only allowed to appear once in a term. Notice that $\widetilde{x}$ can be empty: $M[\leftarrow x]$ expresses that x does not share any variables in M. As in $\lambda^{\natural}_{\oplus}$, the term $\mathtt{fail}^{\widetilde{x}}$ explicitly accounts for failed attempts at substituting the variables $\widetilde{x}$, due to an excess or lack of resources. There is a difference with respect to $\lambda^{\natural}_{\oplus}$: in the term $\mathtt{fail}^{\widetilde{x}}$, $\widetilde{x}$ denotes a set (rather than a multiset) of variables, which may include shared variables.

In $M[\widetilde{x} \leftarrow x]$ we require that (i) every $x_i \in \widetilde{x}$ must occur exactly once in M and that (ii) x_i is not a sharing variable. The occurrence of x_i can appear within the fail term $\mathtt{fail}^{\widetilde{y}}$, if $x_i \in \widetilde{y}$. In the explicit linear substitution $M\langle\!|N/x|\!\rangle$, we require: (i) the variable x has to occur in M; (ii) x cannot be a sharing variable; and (iii) x cannot be in an explicit linear substitution occurring in M. For instance, $M'\langle\!|L/x|\!\rangle\langle\!|N/x|\!\rangle$ is not a valid term in $\widehat{\lambda}^{\natural}_{\oplus}$.

To define the reduction semantics of $\widehat{\lambda}^{\natural}_{\oplus}$, we require some auxiliary notions: the free variables of an expression/term, the head of a term, and linear head substitution.

$$\begin{array}{lll}
\mathsf{fv}(x) = \{x\} & \mathsf{fv}(\mathtt{fail}^{\widetilde{x}}) = \{\widetilde{x}\} & \mathsf{fv}(\wr M \wr) = \mathsf{fv}(M) \\
\mathsf{fv}(B_1 \cdot B_2) = \mathsf{fv}(B_1) \cup \mathsf{fv}(B_2) & \mathsf{fv}(M\ B) = \mathsf{fv}(M) \cup \mathsf{fv}(B) & \mathsf{fv}(\mathtt{1}) = \emptyset \\
\\
\mathsf{fv}(M\langle\!|N/x|\!\rangle) = (\mathsf{fv}(M) \setminus \{x\}) \cup \mathsf{fv}(N) & \mathsf{fv}(M[\widetilde{x} \leftarrow x]) = (\mathsf{fv}(M) \setminus \{\widetilde{x}\}) \cup \{x\} & \\
\mathsf{fv}(\lambda x.(M[\widetilde{x} \leftarrow x])) = \mathsf{fv}(M[\widetilde{x} \leftarrow x]) \setminus \{x\} & \mathsf{fv}(\mathbb{M} + \mathbb{N}) = \mathsf{fv}(\mathbb{M}) \cup \mathsf{fv}(\mathbb{N}) & \\
\\
\mathsf{fv}((M[\widetilde{x} \leftarrow x])\langle\!\langle B/x \rangle\!\rangle) = (\mathsf{fv}(M[\widetilde{x} \leftarrow x]) \setminus \{x\}) \cup \mathsf{fv}(B) &&
\end{array}$$

Figure 1 Free variables for $\widehat{\lambda}^{!}_{\oplus}$.

▶ **Definition 3** (Free Variables). *The set of free variables of a term, bag and expressions in $\widehat{\lambda}^{!}_{\oplus}$, is defined in Fig. 1. As usual, a term M is* closed *if* $\mathsf{fv}(M) = \emptyset$.

▶ **Notation 4.** *We write* $\mathsf{PER}(B)$ *to denote the set of all permutations of bag B. Also, $B_i(n)$ denotes the n-th term in the (permuted) B_i. We define* $\mathsf{size}(B)$ *to denote the number of terms in bag B. That is,* $\mathsf{size}(\mathtt{1}) = 0$ *and* $\mathsf{size}(\wr M \wr \cdot B) = 1 + \mathsf{size}(B)$.

▶ **Definition 5** (Head). *The head of a term M, denoted* $\mathsf{head}(M)$*, is defined inductively:*

$$\begin{array}{ll}
\mathsf{head}(x) = x & \mathsf{head}(\lambda x.(M[\widetilde{x} \leftarrow x])) = \lambda x.(M[\widetilde{x} \leftarrow x]) \\
\mathsf{head}(M\ B) = \mathsf{head}(M) & \mathsf{head}(M\langle\!|N/x|\!\rangle) = \mathsf{head}(M) \\
\mathsf{head}(\mathtt{fail}^{\widetilde{x}}) = \mathtt{fail}^{\widetilde{x}} &
\end{array}$$

$$\mathsf{head}(M[\widetilde{x} \leftarrow x]) = \begin{cases} x & \text{If } \mathsf{head}(M) = y \text{ and } y \in \widetilde{x} \\ \mathsf{head}(M) & \text{Otherwise} \end{cases}$$

$$\mathsf{head}((M[\widetilde{x} \leftarrow x])\langle\!\langle B/x \rangle\!\rangle) = \begin{cases} \mathtt{fail}^{\emptyset} & \text{If } |\widetilde{x}| \neq \mathsf{size}(B) \\ \mathsf{head}(M[\widetilde{x} \leftarrow x]) & \text{Otherwise} \end{cases}$$

▶ **Definition 6** (Linear Head Substitution). *Given a term M with* $\mathsf{head}(M) = x$*, the linear substitution of a term N for x in M, written $M\{\!|N/x|\!\}$ is inductively defined as:*

$$\begin{array}{rll}
x\{\!|N/x|\!\} = N & & \\
(M\ B)\{\!|N/x|\!\} = (M\{\!|N/x|\!\})\ B & & \\
(M\langle\!|L/y|\!\rangle)\{\!|N/x|\!\} = (M\{\!|N/x|\!\})\ \langle\!|L/y|\!\rangle & & x \neq y \\
((M[\widetilde{y} \leftarrow y])\langle\!\langle B/y \rangle\!\rangle)\{\!|N/x|\!\} = (M[\widetilde{y} \leftarrow y]\{\!|N/x|\!\})\ \langle\!\langle B/y \rangle\!\rangle & & x \neq y \\
(M[\widetilde{y} \leftarrow y])\{\!|N/x|\!\} = (M\{\!|N/x|\!\})[\widetilde{y} \leftarrow y] & & x \neq y
\end{array}$$

We now define contexts for terms and expressions in $\widehat{\lambda}^{!}_{\oplus}$. Term contexts involve an explicit linear substitution, rather than an explicit substitution: this is due to the reduction strategy we have chosen to adopt, as we always wish to evaluate explicit substitutions first. Expression contexts can be seen as sums with holes. We assume that the terms that fill in the holes respect the conditions on explicit linear substitutions (i.e., variables appear in a term only once, shared variables must occur in the context).

▶ **Definition 7** (Term and Expression Contexts in $\widehat{\lambda}^{!}_{\oplus}$). *Let $[\cdot]$ denote a hole. Contexts for terms and expressions are defined by the following grammar:*

$$\begin{array}{ll}
C[\cdot], C'[\cdot] & ::= ([\cdot])B \mid ([\cdot])\langle\!|N/x|\!\rangle \mid ([\cdot])[\widetilde{x} \leftarrow x] \mid ([\cdot])[\leftarrow x]\langle\!\langle \mathtt{1}/x \rangle\!\rangle \\
D[\cdot], D'[\cdot] & ::= M + [\cdot] \mid [\cdot] + M
\end{array}$$

The substitution of a hole with term M in a context $C[\cdot]$, denoted $C[M]$, must be a $\widehat{\lambda}^{!}_{\oplus}$-term.

$$[\texttt{RS:Beta}]\ \frac{}{(\lambda x.M[\widetilde{x} \leftarrow x])B \longrightarrow M[\widetilde{x} \leftarrow x]\langle\!\langle B/x \rangle\!\rangle}$$

$$[\texttt{RS:Ex-Sub}]\ \frac{B = \wr M_1 \int \cdots \wr M_k \int \qquad k \geq 1 \qquad M \neq \texttt{fail}^{\widetilde{y}}}{M[x_1,\ldots,x_k \leftarrow x]\langle\!\langle B/x \rangle\!\rangle \longrightarrow \sum_{B_i \in \mathsf{PER}(B)} M\langle\!|B_i(1)/x_1|\!\rangle \cdots \langle\!|B_i(k)/x_k|\!\rangle}$$

$$[\texttt{RS:Lin-Fetch}]\ \frac{\mathsf{head}(M) = x}{M\langle\!|N/x|\!\rangle \longrightarrow M\{\!|N/x|\!\}}$$

$$[\texttt{RS:Fail}]\ \frac{k \neq \mathsf{size}(B) \quad \widetilde{y} = (\mathsf{fv}(M) \setminus \{x_1,\ldots,x_k\}) \cup \mathsf{fv}(B)}{M[x_1,\ldots,x_k \leftarrow x]\langle\!\langle B/x \rangle\!\rangle \longrightarrow \sum_{\mathsf{PER}(B)} \texttt{fail}^{\widetilde{y}}}$$

$$[\texttt{RS:Cons}_1]\ \frac{\widetilde{y} = \mathsf{fv}(B)}{\texttt{fail}^{\widetilde{x}}\ B \longrightarrow \sum_{\mathsf{PER}(B)} \texttt{fail}^{\widetilde{x} \cup \widetilde{y}}} \qquad [\texttt{RS:Cons}_2]\ \frac{\mathsf{size}(B) = k \quad k + |\widetilde{x}| \neq 0 \quad \widetilde{z} = \mathsf{fv}(B)}{(\texttt{fail}^{\widetilde{x} \cup \widetilde{y}}[\widetilde{x} \leftarrow x])\langle\!\langle B/x \rangle\!\rangle \longrightarrow \sum_{\mathsf{PER}(B)} \texttt{fail}^{\widetilde{y} \cup \widetilde{z}}}$$

$$[\texttt{RS:Cons}_3]\ \frac{\widetilde{z} = \mathsf{fv}(N)}{\texttt{fail}^{\widetilde{y} \cup x}\langle\!|N/x|\!\rangle \longrightarrow \texttt{fail}^{\widetilde{y} \cup \widetilde{z}}}$$

$$[\texttt{RS:TCont}]\ \frac{M \longrightarrow M'_1 + \cdots + M'_k}{C[M] \longrightarrow C[M'_1] + \cdots + C[M'_k]} \qquad [\texttt{RS:ECont}]\ \frac{\mathbb{M} \longrightarrow \mathbb{M}'}{D[\mathbb{M}] \longrightarrow D[\mathbb{M}']}$$

Figure 2 Reduction rules for $\widehat{\lambda}^{\ell}_{\oplus}$.

This way, e.g., the hole in context $C[\cdot] = ([\cdot])\langle\!|N/x|\!\rangle$ cannot be filled with y, since $C[y] = (y)\langle\!|N/x|\!\rangle$ is not a well-defined term. Indeed, $M\langle\!|N/x|\!\rangle$ requires that x occurs exactly once within M. Similarly, we cannot fill the hole with $\texttt{fail}^z$ with $z \neq x$, since $C[\texttt{fail}^z] = (\texttt{fail}^z)\langle\!|N/x|\!\rangle$ is also not a well-defined term, for the same reason.

Reduction Semantics

The reduction relation $\longrightarrow$ operates lazily on expressions; it is defined by the rules in Fig. 2. A β-reduction in $\widehat{\lambda}^{\ell}_{\oplus}$ results into an explicit substitution $\langle\!\langle B/x \rangle\!\rangle$, which then evolves into a linear head substitution $\{\!|N/x|\!\}$ (with $N \in B$). Reduction in $\widehat{\lambda}^{\ell}_{\oplus}$ introduces an intermediate step whereby the explicit substitution expands into a sum of terms involving explicit linear substitutions $\langle\!|N/x|\!\rangle$, which are the ones to reduce into a linear head substitution. In the case there is a mismatch between the size of B and the number of shared variables to be substituted, the term reduces to failure.

More specifically, Rule [RS:Beta] is standard and results into an explicit substitution. Rule [RS:Ex-Sub] applies when the size k of the bag coincides with the length of $\widetilde{x} = x_1,\ldots,x_k$. Intuitively, this rule "distributes" an explicit substitution into a sum of terms involving explicit linear substitutions; it considers all possible permutations of the elements in the bag among all shared variables. Rule [RS:Lin-Fetch] specifies the evaluation of a term with an explicit linear substitution into a linear head substitution.

There are three rules reduce to the failure term: their objective is to accumulate all (free) variables involved in failed reductions. Accordingly, Rule [RS:Fail] formalizes failure in the evaluation of an explicit substitution $M[\widetilde{x} \leftarrow x]\langle\!\langle B/x \rangle\!\rangle$, which occurs if there is a mismatch between the resources (terms) present in B and the number of occurrences of

$$
\begin{array}{rl}
M[\leftarrow x]\langle\!\langle \mathtt{1}/x \rangle\!\rangle \succeq_\lambda M & \\
MB\langle\!|N/x|\!\rangle \equiv_\lambda (M\langle\!|N/x|\!\rangle)B & \text{with } x \notin \mathsf{fv}(B) \\
M\langle\!|N_2/y|\!\rangle\langle\!|N_1/x|\!\rangle \equiv_\lambda M\langle\!|N_1/x|\!\rangle\langle\!|N_2/y|\!\rangle & \text{with } x \notin \mathsf{fv}(N_2) \\
MA[\widetilde{x} \leftarrow x]\langle\!\langle B/x \rangle\!\rangle \equiv_\lambda (M[\widetilde{x} \leftarrow x]\langle\!\langle B/x \rangle\!\rangle)A & \text{with } x_i \in \widetilde{x} \Rightarrow x_i \notin \mathsf{fv}(A) \\
M[\widetilde{y} \leftarrow y]\langle\!\langle A/y \rangle\!\rangle[\widetilde{x} \leftarrow x]\langle\!\langle B/x \rangle\!\rangle \succeq_\lambda & \\
\quad (M[\widetilde{x} \leftarrow x]\langle\!\langle B/x \rangle\!\rangle)[\widetilde{y} \leftarrow y]\langle\!\langle A/y \rangle\!\rangle & \text{with } x_i \in \widetilde{x} \Rightarrow x_i \notin \mathsf{fv}(A) \\
C[M] \succeq_\lambda C[M'] & \text{with } M \succeq_\lambda M' \\
D[\mathbb{M}] \succeq_\lambda D[\mathbb{M}'] & \text{with } \mathbb{M} \succeq_\lambda \mathbb{M}'
\end{array}
$$

Figure 3 Precongruence in $\widehat{\lambda}^{\mathsf{f}}_{\oplus}$.

x to be substituted. The resulting failure term preserves all free variables in M and B within its attached set $\widetilde{y}$. Rules [RS:Cons$_1$] and [RS:Cons$_2$] describe reductions that lazily consume the failure term, when a term has $\mathtt{fail}^{\widetilde{x}}$ at its head position. The former rule consumes bags attached to it whilst preserving all its free variables. Finally, Rule [RS:Cons$_3$] accumulates into the failure term the free variables involved in an explicit linear substitution. The contextual rules [RS:TCont] and [RS:Econt] are standard.

▶ **Notation 8.** *As standard, $\longrightarrow$ denotes one step reduction; $\longrightarrow^+$ and $\longrightarrow^*$ denote the transitive and the reflexive-transitive closure of $\longrightarrow$, respectively. We write $\mathbb{N} \longrightarrow_{[\mathtt{R}]} \mathbb{M}$ to denote that [R] is the last (non-contextual) rule used in inferring the step from $\mathbb{N}$ to $\mathbb{M}$.*

▶ **Example 9.** We show how a term can reduce using Rule [RS:Cons$_2$].

$$
\begin{array}{l}
(\lambda x.x_1[x_1 \leftarrow x])\wr \mathtt{fail}^{\emptyset}[\leftarrow y]\langle\!\langle \wr N \wr/y \rangle\!\rangle \wr \longrightarrow_{[\text{RS:Beta}]} x_1[x_1 \leftarrow x]\langle\!\langle \wr \mathtt{fail}^{\emptyset}[\leftarrow y]\langle\!\langle \wr N \wr/y \rangle\!\rangle \wr/x \rangle\!\rangle \\
\quad \longrightarrow_{[\text{RS:Ex-Sub}]} x_1\langle\!|\mathtt{fail}^{\emptyset}[\leftarrow y]\langle\!\langle \wr N \wr/y \rangle\!\rangle/x_1|\!\rangle \longrightarrow_{[\text{RS:Lin-Fetch}]} \mathtt{fail}^{\emptyset}[\leftarrow y]\langle\!\langle \wr N \wr/y \rangle\!\rangle \longrightarrow_{[\text{RS:Cons}_2]} \mathtt{fail}^{\mathsf{fv}(N)}
\end{array}
$$

⌟

Notice that the left-hand sides of the reduction rules in $\widehat{\lambda}^{\mathsf{f}}_{\oplus}$ do not interfere with each other. Reduction in $\widehat{\lambda}^{\mathsf{f}}_{\oplus}$ satisfies a *diamond property*; see [22].

A Precongruence

Fig. 3 defines a precongruence for $\widehat{\lambda}^{\mathsf{f}}_{\oplus}$ on terms and expressions, denoted $\succeq_\lambda$. We write $M \equiv_\lambda M'$ whenever both $M \succeq_\lambda M'$ and $M' \succeq_\lambda M$ hold.

▶ **Example 10.** We illustrate the precongruence in case of failure:

$$
\begin{array}{l}
(\lambda x.x_1[x_1 \leftarrow x])\wr \mathtt{fail}^{\emptyset}[\leftarrow y]\langle\!\langle \mathtt{1}/y \rangle\!\rangle \wr \longrightarrow_{[\text{RS:Beta}]} x_1[x_1 \leftarrow x]\langle\!\langle \wr \mathtt{fail}^{\emptyset}[\leftarrow y]\langle\!\langle \mathtt{1}/y \rangle\!\rangle \wr/x \rangle\!\rangle \\
\quad \longrightarrow_{[\text{RS:Ex-Sub}]} x_1\langle\!|\mathtt{fail}^{\emptyset}[\leftarrow y]\langle\!\langle \mathtt{1}/y \rangle\!\rangle/x_1|\!\rangle \longrightarrow_{[\text{RS:Lin-Fetch}]} \mathtt{fail}^{\emptyset}[\leftarrow y]\langle\!\langle \mathtt{1}/y \rangle\!\rangle \succeq_\lambda \mathtt{fail}^{\emptyset}
\end{array}
$$

In the last step, Rule [RS:Cons$_2$] cannot be applied: y is sharing with no shared variables and the explicit substitution involves the bag $\mathtt{1}$. ⌟

▶ **Example 11.** We illustrate how Rule [RS:Fail] can introduce $\mathtt{fail}^{\widetilde{x}}$ into a term. It also shows how Rule [RS:Cons$_3$] consumes an explicit linear substitution:

$$
\begin{array}{l}
x_1[\leftarrow y]\langle\!\langle \wr N \wr/y \rangle\!\rangle[x_1 \leftarrow x]\langle\!\langle \wr M \wr/x \rangle\!\rangle \longrightarrow_{[\text{RS:Ex-Sub}]} x_1[\leftarrow y]\langle\!\langle \wr N \wr/y \rangle\!\rangle\langle\!|M/x_1|\!\rangle \\
\quad \longrightarrow_{[\text{RS:Fail}]} \mathtt{fail}^{\{x_1\}\cup\mathsf{fv}(N)}\langle\!|M/x_1|\!\rangle \longrightarrow_{[\text{RS:Cons}_3]} \mathtt{fail}^{\mathsf{fv}(M)\cup\mathsf{fv}(N)}
\end{array}
$$

⌟

Intersection Types

We define a type system for $\widehat{\lambda}^{\natural}_{\oplus}$ based on non-idempotent intersection types, similar to the one defined by Bucciarelli et al. in [5]. Intersection types allow us to reason about types of resources in bags but also about every occurrence of a variable. That is, non-idempotent intersection types enable us to distinguish expressions not only by measuring the size of a bag but also by counting the number of times a variable occurs within a term.

▶ **Definition 12** (Types for $\widehat{\lambda}^{\natural}_{\oplus}$). *We define* strict *and* multiset types *by the grammar:*

$$\textit{(Strict)} \quad \sigma, \tau, \delta ::= \mathbf{unit} \mid \pi \to \sigma \qquad\qquad \textit{(Multiset)} \quad \pi, \zeta ::= \bigwedge_{i \in I} \sigma_i \mid \omega$$

A strict type can be the unit type **unit** or a functional type $\pi \to \sigma$, where π is a multiset type and σ is a strict type. Multiset types can be either the empty type ω or an intersection of strict types $\bigwedge_{i \in I} \sigma_i$, with I non-empty. The operator $\wedge$ is commutative, associative, and non-idempotent, that is, $\sigma \wedge \sigma \neq \sigma$. The empty type is the type of the empty bag and acts as the identity element to $\wedge$.

Type assignments range over $\Gamma, \Delta, \ldots$ and have the form $\Gamma, x : \sigma$, assigning the empty type to all but a finite number of variables. Multiple occurrences of a variable can occur within an assignment; they are assigned only strict types. For instance, $x : \tau \to \tau, x : \tau$ is a valid type assignment: it means that x can be of both type $\tau \to \tau$ and τ. The multiset of variables in Γ is denoted as $\mathtt{dom}(\Gamma)$. *Type judgements* are of the form $\Gamma \vdash \mathbb{M} : \sigma$, where Γ consists of variable type assignments, and $\mathbb{M} : \sigma$ means that $\mathbb{M}$ has type σ. We write $\vdash \mathbb{M} : \sigma$ to denote $\emptyset \vdash \mathbb{M} : \sigma$.

▶ **Notation 13.** *Given $k \geq 0$, we shall write σ^k to stand for $\sigma \wedge \cdots \wedge \sigma$ (k times, if $k > 0$) or for ω (if $k = 0$). Similarly, we write $\hat{x} : \sigma^k$ to stand for $x : \sigma, \cdots, x : \sigma$ (k times, if $k > 0$) or for $x : \omega$ (if $k = 0$).*

We define *well-formed* $\widehat{\lambda}^{\natural}_{\oplus}$ expressions, in two stages. We first consider the type system given in Fig. 4 for $\widehat{\lambda}_{\oplus}$, the sub-calculus of $\widehat{\lambda}^{\natural}_{\oplus}$ *without* the failure term $\mathtt{fail}^{\widetilde{x}}$. Then, we define well-formed expressions for the full language $\widehat{\lambda}^{\natural}_{\oplus}$ via Def. 14 (see below).

We first discuss selected rules of the type system for $\widehat{\lambda}_{\oplus}$, which takes into account the sharing construct $M[\widetilde{x} \leftarrow x]$. Rule [TS:var] is standard. Rule [TS:1] assigns the empty bag 1 the empty type ω. The weakening rule [TS:weak] deals with $k = 0$, typing the term $M[\leftarrow x]$, when there are no occurrences of x in M, as long as M is typable. Rule [TS:abs-sh] is as expected: it requires that the sharing variable is assigned the k-fold intersection type σ^k (Not. 13). Rule [TS:app] is standard, requiring a match on the multiset type π. Rule [TS:bag] types the concatenation of bags. Rule [TS:ex-lin-sub] supports explicit linear substitutions. Rule [TS:ex-sub] types explicit substitutions where a bag must consist of both the same type and length of the shared variable it is being substituted for. Rule [TS:sum] types the sum of two expressions of the same type. Rule [TS:share] requires that the shared variables $x_1, \ldots, x_k$ have the same type as the sharing variable x, for $k \neq 0$.

On top of this type system for $\widehat{\lambda}_{\oplus}$, we define well-formed expressions: $\lambda^{\natural}_{\oplus}$-terms whose computation may lead to failure.

▶ **Definition 14** (Well-formedness in $\widehat{\lambda}^{\natural}_{\oplus}$). *An expression $\mathbb{M}$ is well formed if there exist Γ and τ such that $\Gamma \models \mathbb{M} : \tau$ is entailed via the rules in Fig. 5.*

Rules [FS:wf-expr] and [FS:wf-bag] guarantee that every well-typed expression and bag, respectively, is well-formed. Since our language is expressive enough to account for failing computations, we include rules for checking the structure of these ill-behaved terms – terms

$$[\texttt{TS:var}]\ \frac{}{x:\sigma \vdash x:\sigma} \qquad [\texttt{TS:1}]\ \frac{}{\vdash 1:\omega} \qquad [\texttt{TS:weak}]\ \frac{\Delta \vdash M:\tau}{\Delta, x:\omega \vdash M[\leftarrow x]:\tau}$$

$$[\texttt{TS:abs-sh}]\ \frac{\Delta, x:\sigma^k \vdash M[\widetilde{x} \leftarrow x]:\tau}{\Delta \vdash \lambda x.(M[\widetilde{x} \leftarrow x]):\sigma^k \to \tau} \qquad [\texttt{TS:app}]\ \frac{\Gamma \vdash M:\pi \to \tau \qquad \Delta \vdash B:\pi}{\Gamma, \Delta \vdash M\ B:\tau}$$

$$[\texttt{TS:bag}]\ \frac{\Gamma \vdash M:\sigma \qquad \Delta \vdash B:\sigma^k}{\Gamma, \Delta \vdash \wr M \wr \cdot B:\sigma^{k+1}} \qquad [\texttt{TS:ex-lin-sub}]\ \frac{\Delta \vdash N:\sigma \qquad \Gamma, x:\sigma \vdash M:\tau}{\Gamma, \Delta \vdash M\langle\!| N/x |\!\rangle:\tau}$$

$$[\texttt{TS: ex-sub}]\ \frac{\Delta \vdash B:\pi \qquad \Gamma, x:\pi \vdash M[\widetilde{x} \leftarrow x]:\tau}{\Gamma, \Delta \vdash M[\widetilde{x} \leftarrow x]\langle\!\langle B/x \rangle\!\rangle:\tau} \qquad [\texttt{TS:sum}]\ \frac{\Gamma \vdash \mathbb{M}:\sigma \qquad \Gamma \vdash \mathbb{N}:\sigma}{\Gamma \vdash \mathbb{M}+\mathbb{N}:\sigma}$$

$$[\texttt{TS:share}]\ \frac{\Delta, x_1:\sigma, \cdots, x_k:\sigma \vdash M:\tau \quad x \notin \mathtt{dom}(\Delta) \quad k \neq 0}{\Delta, x:\sigma^k \vdash M[x_1, \cdots, x_k \leftarrow x]:\tau}$$

Figure 4 Typing rules for $\widehat{\lambda}_{\oplus}$.

$$[\texttt{FS:wf-expr}]\ \frac{\Gamma \vdash \mathbb{M}:\tau}{\Gamma \models \mathbb{M}:\tau} \qquad [\texttt{FS:wf-bag}]\ \frac{\Gamma \vdash B:\pi}{\Gamma \models B:\pi} \qquad [\texttt{FS:weak}]\ \frac{\Gamma \models M:\tau}{\Gamma, x:\omega \models M[\leftarrow x]:\tau}$$

$$[\texttt{FS:abs-sh}]\ \frac{\Gamma, x:\sigma^k \models M[\widetilde{x} \leftarrow x]:\tau \quad x \notin \mathtt{dom}(\Gamma)}{\Gamma \models \lambda x.(M[\widetilde{x} \leftarrow x]):\sigma^k \to \tau} \qquad [\texttt{FS:fail}]\ \frac{\mathtt{dom}(\Gamma) = \widetilde{x}}{\Gamma \models \mathtt{fail}^{\widetilde{x}}:\tau}$$

$$[\texttt{FS:app}]\ \frac{\Gamma \models M:\sigma^j \to \tau \quad \Delta \models B:\sigma^k}{\Gamma, \Delta \models M\ B:\tau} \qquad [\texttt{FS:bag}]\ \frac{\Gamma \models M:\sigma \quad \Delta \models B:\sigma^k}{\Gamma, \Delta \models \wr M \wr \cdot B:\sigma^{k+1}}$$

$$[\texttt{FS:ex-lin-sub}]\ \frac{\Gamma, x:\sigma \models M:\tau \quad \Delta \models N:\sigma}{\Gamma, \Delta \models M\langle\!| N/x |\!\rangle:\tau} \qquad [\texttt{FS:sum}]\ \frac{\Gamma \models \mathbb{M}:\sigma \quad \Gamma \models \mathbb{N}:\sigma}{\Gamma \models \mathbb{M}+\mathbb{N}:\sigma}$$

$$[\texttt{FS:ex-sub}]\ \frac{\Gamma, x:\sigma^k \models M[\widetilde{x} \leftarrow x]:\tau \qquad \Delta \models B:\sigma^j}{\Gamma, \Delta \models M[\widetilde{x} \leftarrow x]\langle\!\langle B/x \rangle\!\rangle:\tau}$$

$$[\texttt{FS:share}]\ \frac{\Gamma, x_1:\sigma, \cdots, x_k:\sigma \models M:\tau \quad x \notin \mathtt{dom}(\Gamma) \quad k \neq 0}{\Gamma, x:\sigma^k \models M[x_1, \cdots, x_k \leftarrow x]:\tau}$$

Figure 5 Well-formedness rules for $\widehat{\lambda}^{\sharp}_{\oplus}$.

that can be well-formed, but not typable. For instance, Rules [FS:ex-sub] and [FS:app] differ from similar typing rules in Fig. 4: the size of the bags (as declared in their types) is no longer required to match. Also, Rule [FS:fail] has no analogue in the type system: we allow the failure term $\mathtt{fail}^{\widetilde{x}}$ to be well-formed with any type, provided that the context contains the types of the variables in $\widetilde{x}$. The other rules are self-explanatory.

Well-formed expressions satisfy subject reduction (SR); the proof is standard (cf. [22]).

▶ **Theorem 15** (SR in $\widehat{\lambda}^{\sharp}_{\oplus}$). *If* $\Gamma \models \mathbb{M}:\tau$ *and* $\mathbb{M} \longrightarrow \mathbb{M}'$ *then* $\Gamma \models \mathbb{M}':\tau$.

3 sπ: A Session-Typed π-Calculus

The π-calculus [19] is a model of concurrency in which *processes* interact via *names* (or *channels*) to exchange values, which can be themselves names. Here we overview $\mathtt{s}\pi$, introduced by Caires and Pérez in [6], in which *session types* [14, 15] ensure that the two endpoints of a channel perform matching actions: when one endpoint sends, the other receives; when an endpoint closes, the other closes too. Following [7, 27], $\mathtt{s}\pi$ defines a Curry-Howard correspondence between session types and a linear logic with two dual modalities ($\&A$ and $\oplus A$), which define *non-deterministic* sessions. In $\mathtt{s}\pi$, cut elimination corresponds to process communication, proofs correspond to processes, and propositions correspond to session types.

Syntax and Semantics

We use $x, y, z, w \ldots$ to denote names implementing the *(session) endpoints* of protocols specified by session types. We consider the sub-language of [6] without labeled choices and replication, which is actually sufficient to encode $\widehat{\lambda}^{ij}_{\oplus}$.

▶ **Definition 16** (Processes). *The syntax of* $\mathtt{s}\pi$ *processes is given by the grammar:*

$$P, Q ::= \overline{x}(y).P \mid x(y).P \mid x.\overline{\mathtt{close}} \mid x.\mathtt{close}; P \mid [x \leftrightarrow y] \mid (P \mid Q) \mid (\nu x)P \mid \mathbf{0}$$
$$\mid x.\overline{\mathtt{some}}; P \mid x.\overline{\mathtt{none}} \mid x.\mathtt{some}_{(w_1,\cdots,w_n)}; P \mid P \oplus Q$$

In the first line, an output process $\overline{x}(y).P$ sends a fresh name y along session x and then continues as P. An input process $x(y).P$ receives a name z along x and then continues as $P\{^z/_y\}$, which denotes the capture-avoiding substitution of z for y in P. Processes $x.\overline{\mathtt{close}}$ and $x.\mathtt{close}; P$ denote complementary actions for closing session x. The forwarder process $[x \leftrightarrow y]$ denotes a bi-directional link between sessions x and y. Process $P \mid Q$ denotes the parallel execution of P and Q. Process $(\nu x)P$ denotes the process P in which name x has been restricted, i.e., x is kept private to P. $\mathbf{0}$ is the inactive process.

The constructs in the second line introduce non-deterministic sessions which, intuitively, *may* provide a session protocol *or* fail.

- Process $x.\overline{\mathtt{some}}; P$ confirms that the session on x will execute and continues as P. Process $x.\overline{\mathtt{none}}$ signals the failure of implementing the session on x.
- Process $x.\mathtt{some}_{(w_1,\cdots,w_n)}; P$ specifies a dependency on a non-deterministic session x. This process can either (i) synchronize with an action $x.\overline{\mathtt{some}}$ and continue as P, or (ii) synchronize with an action $x.\overline{\mathtt{none}}$, discard P, and propagate the failure on x to $(w_1, \cdots, w_n)$, which are sessions implemented in P. When x is the only session implemented in P, the tuple of dependencies is empty and so we write simply $x.\mathtt{some}; P$.
- $P \oplus Q$ denotes a *non-deterministic choice* between P and Q. We shall often write $\bigoplus_{i \in I} P_i$ to stand for $P_1 \oplus \cdots \oplus P_n$.

In $(\nu y)P$ and $x(y).P$ the distinguished occurrence of name y is binding, with scope P. The set of free names of P is denoted by $fn(P)$.

The *reduction semantics* of $\mathtt{s}\pi$ specifies the computations that a process performs on its own (Fig. 6). It relies on *structural congruence*, denoted $\equiv$, which expresses basic identities on the structure of processes and the non-collapsing nature of non-determinism (cf. [22]).

In Fig. 6, the first reduction rule formalizes communication, which concerns bound names only (internal mobility): name y is bound in both $\overline{x}(y).Q$ and $x(y).P$. The reduction rule for the forwarder process leads to a name substitution. The reduction rule for closing a session is self-explanatory, as is the rule in which prefix $x.\overline{\mathtt{some}}$ confirms the availability of a non-deterministic session. When the non-deterministic session is not available, prefix $x.\overline{\mathtt{none}}$

$$
\begin{array}{rcl}
\overline{x}(y).Q \mid x(y).P & \longrightarrow & (\nu y)(Q \mid P) \\
(\nu x)([x \leftrightarrow y] \mid P) & \longrightarrow & P\{y/x\} \quad (x \neq y) \\
x.\overline{\texttt{close}} \mid x.\texttt{close}; P & \longrightarrow & P \\
x.\overline{\texttt{some}}; P \mid x.\texttt{some}_{(w_1,\cdots,w_n)}; Q & \longrightarrow & P \mid Q \\
x.\overline{\texttt{none}} \mid x.\texttt{some}_{(w_1,\cdots,w_n)}; Q & \longrightarrow & w_1.\overline{\texttt{none}} \mid \cdots \mid w_n.\overline{\texttt{none}}
\end{array}
$$

$$
\begin{array}{ll}
P \equiv P' \wedge P' \longrightarrow Q' \wedge Q' \equiv Q \Rightarrow P \longrightarrow Q & Q \longrightarrow Q' \Rightarrow P \mid Q \longrightarrow P \mid Q' \\
P \longrightarrow Q \Rightarrow (\nu y)P \longrightarrow (\nu y)Q & Q \longrightarrow Q' \Rightarrow P \oplus Q \longrightarrow P \oplus Q'
\end{array}
$$

Figure 6 Reduction for $\mathsf{s}\pi$.

triggers this failure to all dependent sessions $w_1, \ldots, w_n$; this may in turn trigger further failures (i.e., on sessions that depend on $w_1, \ldots, w_n$). Reduction is closed under structural congruence. The remaining rules define contextual reduction with respect to restriction, parallel composition, and non-deterministic choice.

Type System

We introduce the session types that govern process behavior:

▶ **Definition 17** (Session Types). *Session types are given by*

$$A, B ::= \bot \mid \mathbf{1} \mid A \otimes B \mid A \parr B \mid \&A \mid \oplus A$$

Types are assigned to names: an *assignment* $x : A$ enforces the use of name x according to the protocol specified by A. The multiplicative units $\bot$ and $\mathbf{1}$ are used to type terminated (closed) endpoints. We use $A \otimes B$ to type a name that first outputs a name of type A before proceeding as specified by B. Similarly, $A \parr B$ types a name that first inputs a name of type A before proceeding as specified by B. Then we have the two modalities introduced in [6]. We use $\&A$ as the type of a (non-deterministic) session that *may produce* a behavior of type A. Dually, $\oplus A$ denotes the type of a session that *may consume* a behavior of type A.

The two endpoints of a session should be *dual* to ensure absence of communication errors. The dual of a type A is denoted $\overline{A}$. Duality corresponds to negation $(\cdot)^{\perp}$ in linear logic [6]:

▶ **Definition 18** (Duality). *The duality relation on types is given by:*

$$\overline{\mathbf{1}} = \bot \qquad \overline{\bot} = \mathbf{1} \qquad \overline{A \otimes B} = \overline{A} \parr \overline{B} \qquad \overline{A \parr B} = \overline{A} \otimes \overline{B} \qquad \overline{\oplus A} = \&\overline{A} \qquad \overline{\&A} = \oplus\overline{A}$$

Typing judgments are of the form $P \vdash \Delta$, where P is a process and Δ is a linear context of assignments of types to names. The empty context is denoted "$\cdot$". We write $\&\Delta$ to denote that all assignments in Δ have a non-deterministic type, i.e., $\Delta = w_1{:}\&A_1, \ldots, w_n{:}\&A_n$, for some $A_1, \ldots, A_n$. The typing judgment $P \vdash \Delta$ corresponds to the logical sequent $\vdash \Delta$ for classical linear logic, which can be recovered by erasing processes and name assignments.

Typing rules for processes correspond to proof rules in the logic; Fig. 7 gives a selection (see [6] and [22] for a full account). Rule [$\mathtt{Tid}$] interprets the identity axiom using the forwarder process. Rules [$\mathtt{T1}$] and [$\mathtt{T}\bot$] type the constructs for session termination. Rules [$\mathtt{T}\otimes$] and [$\mathtt{T}\parr$] type output and input of a name along a session, respectively. The last four rules are used to type constructs for non-determinism and failure. Rules [$\mathtt{T}\&^{\mathtt{x}}_{\mathtt{d}}$] and [$\mathtt{T}\&^{\mathtt{x}}$] introduce a session of type $\&A$, which may produce a behavior of type A: while the former rule covers the case in which $x : A$ is available, the latter rule formalizes the case in which $x : A$ is not available (i.e., a failure). Given a sequence of names $\widetilde{w} = w_1, \ldots, w_n$, Rule [$\mathtt{T}\oplus^{\mathtt{x}}_{\widetilde{\mathtt{w}}}$] accounts

$$[\mathtt{Tid}]\ \frac{}{[x \leftrightarrow y] \vdash x{:}A, y{:}\overline{A}} \qquad [\mathtt{T1}]\ \frac{}{x.\overline{\mathtt{close}} \vdash x : \mathbf{1}} \qquad [\mathtt{T}\bot]\ \frac{P \vdash \Delta}{x.\mathtt{close}; P \vdash x{:}\bot, \Delta}$$

$$[\mathtt{T}\otimes]\ \frac{P \vdash \Delta, y : A \quad Q \vdash \Delta', x : B}{\overline{x}(y).(P \mid Q) \vdash \Delta, \Delta', x : A \otimes B} \qquad [\mathtt{T}\parr]\ \frac{P \vdash \Delta, y : C, x : D}{x(y).P \vdash \Delta, x : C \parr D} \qquad [\mathtt{T}\&^{\mathtt{x}}]\ \frac{}{x.\overline{\mathtt{none}} \vdash x : \&A}$$

$$[\mathtt{T}\oplus^{\mathtt{x}}_{\mathtt{w}}]\ \frac{P \vdash \widetilde{w} : \&\Delta, x : A}{x.\mathtt{some}_{\widetilde{w}}; P \vdash \widetilde{w}{:}\&\Delta, x{:}\oplus A} \qquad [\mathtt{T}\&^{\mathtt{x}}_{\mathtt{d}}]\ \frac{P \vdash \Delta, x : A}{x.\overline{\mathtt{some}}; P \vdash \Delta, x : \&A} \qquad [\mathtt{T}\&]\ \frac{P \vdash \&\Delta \qquad Q \vdash \&\Delta}{P \oplus Q \vdash \&\Delta}$$

Figure 7 Selected typing rules for $\mathsf{s}\pi$.

for the possibility of not being able to consume the session $x : A$ by considering sessions different from x as potentially not available. Finally, Rule [T&] expresses non-deterministic choice of processes P and Q that implement non-deterministic behaviors only.

The type system enjoys type preservation, a result that follows directly from the cut elimination property in the underlying logic; it ensures that the observable interface of a system is invariant under reduction. The type system also ensures other properties for well-typed processes (e.g. global progress and confluence); see [6] for details.

▶ **Theorem 19** (Type Preservation [6]). *If $P \vdash \Delta$ and $P \longrightarrow Q$ then $Q \vdash \Delta$.*

4 The Encoding

To encode $\lambda^{\natural}_{\oplus}$ into $\mathsf{s}\pi$, we first define the encoding $(\!|\cdot|\!)^{\circ}$ from well-formed expressions in $\lambda^{\natural}_{\oplus}$ to well-formed expressions in $\widehat{\lambda}^{\natural}_{\oplus}$. Then, the encoding $[\![\cdot]\!]^{\natural}_{u}$ (for a name u) translates well-formed expressions in $\widehat{\lambda}^{\natural}_{\oplus}$ to well-typed processes in $\mathsf{s}\pi$. We first discuss the encodability criteria.

4.1 Encodability Criteria

We follow most of the criteria in [11], a widely studied abstract framework for establishing the *quality* of encodings. A *language* $\mathcal{L}$ is a pair: a set of terms and a reduction semantics $\longrightarrow$ on terms (with reflexive, transitive closure denoted $\xrightarrow{*}$). A correct encoding translates terms of a source language $\mathcal{L}_1$ into terms of a target language $\mathcal{L}_2$ by respecting certain criteria. The criteria in [11] concern *untyped* languages; because we treat *typed* languages, we follow [17] in requiring that encodings preserve typability.

▶ **Definition 20** (Correct Encoding). *Let $\mathcal{L}_1 = (\mathcal{M}, \longrightarrow_1)$ and $\mathcal{L}_2 = (\mathcal{P}, \longrightarrow_2)$ be two languages. We use $M, M', \ldots$ and $P, P', \ldots$ to range over elements in $\mathcal{M}$ and $\mathcal{P}$. Also, let $\approx_2$ be a behavioral equivalence on terms in $\mathcal{P}$. We say that a translation $[\![\cdot]\!] : \mathcal{M} \to \mathcal{P}$ is a* correct encoding *if it satisfies the following criteria:*

1. *Type preservation: For every well-typed M, it holds that $[\![M]\!]$ is well-typed.*
2. *Operational Completeness: For every M, M' such that $M \xrightarrow{*}_1 M'$, it holds that $[\![M]\!] \xrightarrow{*}_2 \approx_2 [\![M']\!]$.*
3. *Operational Soundness: For every M and P such that $[\![M]\!] \xrightarrow{*}_2 P$, there exists an M' such that $M \longrightarrow^{*}_1 M'$ and $P \xrightarrow{*}_2 \approx_2 [\![M']\!]$.*
4. *Success Sensitiveness: For every M, it holds that $M{\checkmark}_1$ if and only if $[\![M]\!]{\checkmark}_2$, where ${\checkmark}_1$ and ${\checkmark}_2$ denote a success predicate in $\mathcal{M}$ and $\mathcal{P}$, respectively.*

Besides these semantic criteria, we also consider *compositionality*, a syntactic criterion that requires that a composite source term is encoded as the combination of the encodings of its sub-terms. Operational completeness formalizes how reduction steps of a source term are mimicked by its corresponding encoding in the target language; $\approx_2$ conveniently

abstracts away from target terms useful in the translation but which are not meaningful in comparisons. Operational soundness concerns the opposite direction: it formalizes the correspondence between (i) the reductions of a target term obtained via the translation and (ii) the reductions of the corresponding source term. The role of $\approx_2$ can be explained as in completeness. Success sensitiveness complements completeness and soundness, which concern reductions and therefore do not contain information about observable behaviors. The so-called success predicates $\checkmark_1$ and $\checkmark_2$ serve as a minimal notion of *observables*; the criterion then says that observability of success of a source term implies observability of success in the corresponding target term, and viceversa. Finally, type preservation is self-explanatory.

We choose not to use *full abstraction* as a correctness criterion. As argued in [12], full abstraction is not an informative criterion when it comes to an encoding's quality.

4.2 First Step: From $\lambda^{\lightning}_{\oplus}$ into $\widehat{\lambda}^{\lightning}_{\oplus}$

We define an encoding $(\!|-|\!)^{\circ}$ from $\lambda^{\lightning}_{\oplus}$ into $\widehat{\lambda}^{\lightning}_{\oplus}$ and prove it is correct. The encoding, defined for well-formed terms in $\lambda^{\lightning}_{\oplus}$ (cf. Def. 42 in App. A.1), relies on an intermediate encoding $(\!|\cdot|\!)^{\bullet}$ on closed $\lambda^{\lightning}_{\oplus}$-terms.

We introduce some notation. Given a term M such that $\#(x, M) = k$ and a sequence of pairwise distinct fresh variables $\widetilde{x} = x_1, \ldots, x_k$ we write $M\langle\widetilde{x}/x\rangle$ or $M\langle x_1, \cdots, x_n/x\rangle$ to stand for $M\langle x_1/x\rangle \cdots \langle x_k/x\rangle$. That is, $M\langle\widetilde{x}/x\rangle$ denotes a simultaneous linear substitution whereby each distinct occurrence of x in M is replaced by a distinct $x_i \in \widetilde{x}$. Notice that each x_i has the same type as x. We use (simultaneous) linear substitutions to force all bound variables in $\lambda^{\lightning}_{\oplus}$ to become shared variables in $\widehat{\lambda}^{\lightning}_{\oplus}$.

▶ **Definition 21** (From $\lambda^{\lightning}_{\oplus}$ to $\widehat{\lambda}^{\lightning}_{\oplus}$). *Let $M \in \lambda^{\lightning}_{\oplus}$. Suppose $\Gamma \models M : \tau$, with $\mathtt{dom}(\Gamma) = \mathsf{fv}(M) = \{x_1, \cdots, x_k\}$ and $\#(x_i, M) = j_i$. We define $(\!|M|\!)^{\circ}$ as*

$$(\!|M|\!)^{\circ} = (\!|M\langle\widetilde{x_1}/x_1\rangle \cdots \langle\widetilde{x_k}/x_k\rangle|\!)^{\bullet}[\widetilde{x_1} \leftarrow x_1] \cdots [\widetilde{x_k} \leftarrow x_k]$$

where $\widetilde{x_i} = x_{i_1}, \cdots, x_{i_{j_i}}$ and the encoding $(\!|\cdot|\!)^{\bullet} : \lambda^{\lightning}_{\oplus} \to \widehat{\lambda}^{\lightning}_{\oplus}$ is defined in Fig. 8 on closed $\lambda^{\lightning}_{\oplus}$-terms. The encoding $(\!|\cdot|\!)^{\circ}$ extends homomorphically to expressions.

The encoding $(\!|-|\!)^{\circ}$ "atomizes" occurrences of variables: it converts n occurrences of a variable x in a term into n distinct variables $x_1, \ldots, x_n$. The sharing construct coordinates the occurrences of these variables by constraining each to occur exactly once within a term. We proceed in two stages. First, we share all free variables using $(\!|-|\!)^{\circ}$: this ensures that free variables are replaced by bound shared variables. Second, we apply the encoding $(\!|-|\!)^{\bullet}$ on the corresponding closed term. Two cases of Fig. 8 are noteworthy. In $(\!|\lambda x.M|\!)^{\bullet}$, the occurrences of x are replaced with fresh shared variables that only occur once within in M. The definition of $(\!|M\langle\!\langle B/x\rangle\!\rangle|\!)^{\bullet}$ considers two possibilities. If the bag being encoded is non-empty and the explicit substitution would not lead to failure (the number of occurrences of x and the size of the bag coincide) then we encode the explicit substitution as a sum of explicit linear substitutions. Otherwise, the explicit substitution will lead to a failure, and the encoding proceeds inductively. As we will see, doing this will enable a tight operational correspondence result with $\mathsf{s}\pi$.

$$(\!|x|\!)^\bullet = x \quad (\!|\wr M \wr \cdot B|\!)^\bullet = \wr (\!|M|\!)^\bullet \wr \cdot (\!|B|\!)^\bullet \quad (\!|\widetilde{\mathtt{fail}^x}|\!)^\bullet = \widetilde{\mathtt{fail}^x} \quad (\!|M\ B|\!)^\bullet = (\!|M|\!)^\bullet\ (\!|B|\!)^\bullet$$
$$(\!|\mathtt{1}|\!)^\bullet = \mathtt{1} (\!|\lambda x.M|\!)^\bullet = \lambda x.((\!|M\langle \widetilde{x}/x\rangle|\!)^\bullet[\widetilde{x} \leftarrow x]) \quad \#(x,M) = n, \text{ each } x_i \text{ is fresh}$$

$$(\!|M\langle\!\langle B/x\rangle\!\rangle|\!)^\bullet =$$
$$\begin{cases} \sum_{B_i \in \mathsf{PER}((\!|B|\!)^\bullet)} (\!|M\langle \widetilde{x}/x\rangle|\!)^\bullet \langle\!| B_i(1)/x_1 |\!\rangle \cdots \langle\!| B_i(k)/x_k |\!\rangle & \#(x,M) = \mathsf{size}(B) = k \geq 1 \\ (\!|M\langle x_1 . \cdots, x_k/x\rangle|\!)^\bullet[\widetilde{x} \leftarrow x]\langle\!\langle (\!|B|\!)^\bullet/x\rangle\!\rangle & \text{otherwise, } \#(x,M) = k \geq 0 \end{cases}$$

Figure 8 Auxiliary Encoding: $\lambda^{\natural}_{\oplus}$ into $\widehat{\lambda}^{\natural}_{\oplus}$.

▶ **Example 22.** Consider the $\lambda^{\natural}_{\oplus}$ term $y\langle\!\langle B/x\rangle\!\rangle$, with $\mathsf{fv}(B) = \emptyset$ and $y \neq x$. Its encoding into $\widehat{\lambda}^{\natural}_{\oplus}$ is $(\!|y\langle\!\langle B/x\rangle\!\rangle|\!)^\circ = (\!|y_0\langle\!\langle B/x\rangle\!\rangle|\!)^\bullet[y_0 \leftarrow y] = y_0[\leftarrow x]\langle\!\langle (\!|B|\!)^\bullet/x\rangle\!\rangle[y_0 \leftarrow y]$. Notice that the encoding induces (empty) sharing on x, even if x does not occur in the term y. ⌟

We consider correctness (Def. 20) for $(\!|\cdot|\!)^\circ$. Our encoding is in "two-levels", because $(\!|\cdot|\!)^\circ$ it is defined in terms of $(\!|\cdot|\!)^\bullet$. As such, it satisfies a weak form of compositionality [11]. In [22] we have established the following:

▶ **Theorem 23** (Correctness for $(\!|\cdot|\!)^\circ$). *The encoding $(\!|\cdot|\!)^\circ$ is type preserving, operationally complete, operationally sound, and success sensitive.*

4.3 Second Step: From $\widehat{\lambda}^{\natural}_{\oplus}$ to $\mathsf{s}\pi$

We now define our encoding of $\widehat{\lambda}^{\natural}_{\oplus}$ into $\mathsf{s}\pi$, and establish its correctness.

▶ **Definition 24** (From $\widehat{\lambda}^{\natural}_{\oplus}$ into $\mathsf{s}\pi$: Expressions). *Let u be a name. The encoding $[\![\cdot]\!]^{\natural}_u : \widehat{\lambda}^{\natural}_{\oplus} \to \mathsf{s}\pi$ is defined in Fig. 9.*

As usual in encodings of λ into π, we use a name u to provide the behaviour of the encoded expression. Here u is a non-deterministic session: the encoded expression can be available or not; this is signaled by prefixes $u.\overline{\mathtt{some}}$ and $u.\overline{\mathtt{none}}$, respectively. Notice that every (free) variable x in a $\widehat{\lambda}^{\natural}_{\oplus}$ expression becomes a name x in its corresponding $\mathsf{s}\pi$ process.

We discuss the most interesting aspects of the translation in Fig. 9. The term $M\,B$ is encoded into a non-deterministic sum: this models the fact that application involves a choice in the order in which the elements of the bag are substituted. The encoding of $M\langle\!|N/x|\!\rangle$ is the parallel composition of the translations of M and N. We need to wait for confirmation of a behaviour along the variable that is being substituted. The encoding of $M[x_1, \cdots, x_n \leftarrow x]$ first confirms the availability of the behavior along x. Then it sends a dummy variable y_i, which is used to collapse the process in the case of a failed reduction. Subsequently, for each shared variable, the encoding receives a name, which will act as an occurrence of the shared variable. At the end, we use $x.\overline{\mathtt{none}}$ to signal that there is no further information to send over. The encoding of $\wr M \wr \cdot B$ synchronises with the encoding of $M[x_1, \cdots, x_n \leftarrow x]$, just discussed. The name y_i is used to trigger a failure in the computation if there is a lack of elements in the encoding of bag. The encoding of $\mathtt{fail}^{x_1,\cdots,x_k}$ simply triggers failure on u and on each of $x_1, \cdots, x_k$. The encoding of $[\![\mathbb{M} + \mathbb{N}]\!]^{\natural}_u$ homomorphically preserves non-determinism.

$$[\![x]\!]^{\ell}_{u} = x.\overline{\texttt{some}};[x \leftrightarrow u]$$

$$[\![\lambda x.M[\widetilde{x} \leftarrow x]]\!]^{\ell}_{u} = u.\overline{\texttt{some}};u(x).[\![M[\widetilde{x} \leftarrow x]]\!]^{\ell}_{u}$$

$$[\![MB]\!]^{\ell}_{u} = \bigoplus_{B_i \in \mathsf{PER}(B)} (\nu v)([\![M]\!]^{\ell}_{v} \mid v.\texttt{some}_{u,\mathsf{fv}(B)};\overline{v}(x).([v \leftrightarrow u] \mid [\![B_i]\!]^{\ell}_{x}))$$

$$[\![M[\widetilde{x} \leftarrow x]\langle\!\langle B/x \rangle\!\rangle]\!]^{\ell}_{u} = \bigoplus_{B_i \in \mathsf{PER}(B)} (\nu x)([\![M[\widetilde{x} \leftarrow x]]\!]^{\ell}_{u} \mid [\![B_i]\!]^{\ell}_{x})$$

$$[\![M\langle\!\langle N/x \rangle\!\rangle]\!]^{\ell}_{u} = (\nu x)([\![M]\!]^{\ell}_{u} \mid x.\texttt{some}_{\mathsf{fv}(N)};[\![N]\!]^{\ell}_{x})$$

$$[\![M[\leftarrow x]]\!]^{\ell}_{u} = x.\overline{\texttt{some}}.\overline{x}(y_i).(y_i.\texttt{some}_{u,\mathsf{fv}(M)};y_i.\texttt{close};[\![M]\!]^{\ell}_{u} \mid x.\overline{\texttt{none}})$$

$$\begin{aligned}[\![M[x_1,\cdots,x_n \leftarrow x]]\!]^{\ell}_{u} = {}& \\ & x.\overline{\texttt{some}}.\overline{x}(y_1).(y_1.\texttt{some}_{\emptyset};y_1.\texttt{close};\mathbf{0} \\ & \mid x.\overline{\texttt{some}};x.\texttt{some}_{u,(\mathsf{fv}(M)\setminus x_1,\cdots,x_n)};x(x_1).\cdots \\ & .x.\overline{\texttt{some}}.\overline{x}(y_n).(y_n.\texttt{some}_{\emptyset};y_n.\texttt{close};\mathbf{0} \mid x.\overline{\texttt{some}};x.\texttt{some}_{u,(\mathsf{fv}(M)\setminus x_n)};x(x_n) \\ & .x.\overline{\texttt{some}};\overline{x}(y_{n+1}).(y_{n+1}.\texttt{some}_{u,\mathsf{fv}(M)};y_{n+1}.\texttt{close};[\![M]\!]^{\ell}_{u} \mid x.\overline{\texttt{none}})\ )\cdots)\end{aligned}$$

$$[\![\texttt{fail}^{x_1,\cdots,x_k}]\!]^{\ell}_{u} = u.\overline{\texttt{none}} \mid x_1.\overline{\texttt{none}} \mid \cdots \mid x_k.\overline{\texttt{none}}$$

$$[\![\mathtt{1}]\!]^{\ell}_{x} = x.\texttt{some}_{\emptyset};x(y_n).(y_n.\overline{\texttt{some}};y_n.\overline{\texttt{close}} \mid x.\texttt{some}_{\emptyset};x.\overline{\texttt{none}})$$

$$\begin{aligned}[\![\wr M \wr \cdot B]\!]^{\ell}_{x} = {}& x.\texttt{some}_{\mathsf{fv}(\wr M \wr \cdot B)};x(y_i).x.\texttt{some}_{y_i,\mathsf{fv}(\wr M \wr \cdot B)};x.\overline{\texttt{some}};\overline{x}(x_i) \\ & .(x_i.\texttt{some}_{\mathsf{fv}(M)};[\![M]\!]^{\ell}_{x_i} \mid [\![B]\!]^{\ell}_{x} \mid y_i.\overline{\texttt{none}})\end{aligned}$$

$$[\![\mathbb{M}+\mathbb{N}]\!]^{\ell}_{u} = [\![\mathbb{M}]\!]^{\ell}_{u} \oplus [\![\mathbb{N}]\!]^{\ell}_{u}$$

Figure 9 Encoding $\widehat{\lambda}^{\ell}_{\oplus}$ expressions into $\mathsf{s}\pi$ processes.

▶ **Example 25.** We illustrate $[\![\cdot]\!]^{\ell}_{u}$ in Fig. 9 by encoding the $\widehat{\lambda}^{\ell}_{\oplus}$-terms $N[\leftarrow x]\langle\!\langle \wr M \wr/x \rangle\!\rangle$ and $\texttt{fail}^{\mathsf{fv}(N)\cup\mathsf{fv}(M)}$, where M, N are closed well-formed $\widehat{\lambda}^{\ell}_{\oplus}$-terms (i.e. $\mathsf{fv}(N) = \mathsf{fv}(M) = \emptyset$):

$$\begin{aligned}[\![N[\leftarrow x]\langle\!\langle \wr M \wr/x \rangle\!\rangle]\!]^{\ell}_{u} &= (\nu x)([\![N[\leftarrow x]]\!]^{\ell}_{u} \mid [\![\wr M \wr]\!]^{\ell}_{x}) \\ &= (\nu x)(x.\overline{\texttt{some}}.\overline{x}(y_i).(y_i.\texttt{some}_{u};y_i.\texttt{close};[\![N]\!]^{\ell}_{u} \mid x.\overline{\texttt{none}}) \mid \\ &\qquad x.\texttt{some}_{\emptyset};x(y_i).x.\texttt{some}_{y_i};x.\overline{\texttt{some}};\overline{x}(x_i) \\ &\qquad .(x_i.\texttt{some}_{\emptyset};[\![M]\!]^{\ell}_{x_i} \mid [\![\mathtt{1}]\!]^{\ell}_{x} \mid y_i.\overline{\texttt{none}})) \\ [\![\texttt{fail}^{\mathsf{fv}(N)\cup\mathsf{fv}(M)}]\!]^{\ell}_{u} &= u.\overline{\texttt{none}}\end{aligned}$$

⌟

We now encode intersection types (for $\lambda^{\ell}_{\oplus}$ and $\widehat{\lambda}^{\ell}_{\oplus}$) into session types (for $\mathsf{s}\pi$):

▶ **Definition 26** (From $\widehat{\lambda}^{\ell}_{\oplus}$ into $\mathsf{s}\pi$: Types). *The translation $[\![\cdot]\!]^{\ell}$ on types is defined in Fig. 10. Let Γ be an assignment defined as $\Gamma = x_1 : \sigma_1,\cdots,x_m : \sigma_k, v_1 : \pi_1,\cdots,v_n : \pi_n$. We define $[\![\Gamma]\!]^{\ell}$ as $x_1 : \&\overline{[\![\sigma_1]\!]^{\ell}},\cdots,x_k : \&\overline{[\![\sigma_k]\!]^{\ell}}, v_1 : \&\overline{[\![\pi_1]\!]^{\ell}_{(\sigma,i_1)}},\cdots,v_n : \&\overline{[\![\pi_n]\!]^{\ell}_{(\sigma,i_n)}}$.*

The encoding of types captures our use of non-deterministic session protocols (typed with "&") to represent non-deterministic and fail-prone evaluation in $\widehat{\lambda}_{\oplus}$. Notice that the encoding of the multiset type π depends on two arguments (a strict type σ and a number $i \geq 0$) which are left unspecified above. This is crucial to represent mismatches in $\widehat{\lambda}^{\ell}_{\oplus}$ (i.e., sources of failures) as typable processes in $\mathsf{s}\pi$. For instance, in Fig. 5, Rule [FS:app] admits a mismatch between $\sigma^j \to \tau$ and σ^k, for it allows $j \neq k$. In our proof of type preservation, these two arguments are instantiated appropriately, enabling typability as session-typed processes.

$$
\begin{aligned}
[\![\mathbf{unit}]\!]^{\natural} &= \&\mathbf{1} \\
[\![\pi \to \tau]\!]^{\natural} &= \&((\overline{[\![\pi]\!]^{\natural}_{(\sigma,i)}}) \parr [\![\tau]\!]^{\natural}) \quad \text{(for some strict type } \sigma\text{, with } i \geq 0) \\
[\![\sigma \wedge \pi]\!]^{\natural}_{(\sigma,i)} &= \overline{\&((\oplus\bot) \otimes (\& \oplus ((\&\overline{[\![\sigma]\!]^{\natural}}) \parr (\overline{[\![\pi]\!]^{\natural}_{(\sigma,i)}})))} \\
&= \oplus((\&\mathbf{1}) \parr (\oplus\&((\oplus[\![\sigma]\!]^{\natural}) \otimes ([\![\pi]\!]^{\natural}_{(\sigma,i)})))) \\
[\![\omega]\!]^{\natural}_{(\sigma,i)} &= \begin{cases} \overline{\&((\oplus\bot) \otimes (\& \oplus \bot))} & \text{if } i = 0 \\ \overline{\&((\oplus\bot) \otimes (\& \oplus ((\&\overline{[\![\sigma]\!]^{\natural}}) \parr (\overline{[\![\omega]\!]^{\natural}_{(\sigma,i-1)}})))} & \text{if } i > 0 \end{cases}
\end{aligned}
$$

Figure 10 Encoding types for $\widehat{\lambda}^{\natural}_{\oplus}$ as session types.

With our encodings of expressions and types in place, we can now encode judgments:

▶ **Definition 27** (Encoding Judgments). *If* $\Gamma \models \mathbb{M} : \tau$ *then* $[\![\mathbb{M}]\!]^{\natural}_{u} \vdash [\![\Gamma]\!]^{\natural}, u : [\![\tau]\!]^{\natural}$.

We are now ready to consider correctness for $[\![\cdot]\!]^{\natural}$, as in Def. 20. First, the compositionality property follows directly from Fig. 9. We now state the remaining properties in Def. 20, which we have established in [22]. First, type preservation:

▶ **Theorem 28** (Type Preservation for $[\![\cdot]\!]^{\natural}_{u}$). *Let* B *and* $\mathbb{M}$ *be a bag and an expression.*
1. *If* $\Gamma \models B : \pi$ *then* $[\![B]\!]^{\natural}_{u} \models [\![\Gamma]\!]^{\natural}, u : [\![\pi]\!]^{\natural}_{(\sigma,i)}$, *for some strict type* σ *and some* i.
2. *If* $\Gamma \models \mathbb{M} : \tau$ *then* $[\![\mathbb{M}]\!]^{\natural}_{u} \models [\![\Gamma]\!]^{\natural}, u : [\![\tau]\!]^{\natural}$.

We now consider operational completeness. Because $\widehat{\lambda}^{\natural}_{\oplus}$ satisfies the diamond property, it suffices to consider completeness based on a single reduction step ($\mathbb{N} \longrightarrow \mathbb{M}$):

▶ **Theorem 29** (Operational Completeness). *Let* $\mathbb{N}$ *and* $\mathbb{M}$ *be well-formed* $\widehat{\lambda}^{\natural}_{\oplus}$ *closed expressions. If* $\mathbb{N} \longrightarrow \mathbb{M}$ *then there exists* Q *such that* $[\![\mathbb{N}]\!]^{\natural}_{u} \longrightarrow^{*} Q = [\![\mathbb{M}]\!]^{\natural}_{u}$.

▶ **Example 30** (Cont. Example 25). Since M and N are well-formed we can verify, by applying rules in Fig. 5 that, $N[\leftarrow x]\langle\!\langle (\!|M|\!) / x \rangle\!\rangle$ and $\mathtt{fail}^{\mathsf{fv}(N)\cup\mathsf{fv}(M)}$ are well-formed. Notice that $N[\leftarrow x]\langle\!\langle (\!|M|\!) / x \rangle\!\rangle \longrightarrow_{[\mathtt{RS:Fail}]} \mathtt{fail}^{\mathsf{fv}(N)\cup\mathsf{fv}(M)}$. The encoding of the lhs reduces to encoding of the rhs via the reduction rules of $\mathsf{s}\pi$ (Fig. 6) as $[\![N[\leftarrow x]\langle\!\langle (\!|M|\!) / x \rangle\!\rangle]\!]^{\natural}_{u} \longrightarrow^{*} [\![\mathtt{fail}^{\mathsf{fv}(N)\cup\mathsf{fv}(M)}]\!]^{\natural}_{u}$. The complete example with the reduction steps can be found in [22]. ⌟

In soundness we use the precongruence $\succeq_{\lambda}$ (Fig. 3). We write $N \longrightarrow_{\succeq_{\lambda}} N'$ iff $N \succeq_{\lambda} N_1 \longrightarrow N_2 \succeq_{\lambda} N'$, for some N_1, N_2. The reflexive, transitive closure of $\longrightarrow_{\succeq_{\lambda}}$ is $\longrightarrow^{*}_{\succeq_{\lambda}}$.

▶ **Theorem 31** (Operational Soundness). *Let* $\mathbb{N}$ *be a well-formed, closed* $\widehat{\lambda}^{\natural}_{\oplus}$ *expression. If* $[\![\mathbb{N}]\!]^{\natural}_{u} \longrightarrow^{*} Q$ *then* $Q \longrightarrow^{*} Q'$, $\mathbb{N} \longrightarrow^{*}_{\succeq_{\lambda}} \mathbb{N}'$ *and* $[\![\mathbb{N}']\!]^{\natural}_{u} = Q'$, *for some* $Q', \mathbb{N}'$.

Finally, we consider success sensitiveness. This requires extending $\widehat{\lambda}^{\natural}_{\oplus}$ and $\mathsf{s}\pi$ with success predicates. In $\mathsf{s}\pi$, we say that P is unguarded if it does not occur behind a prefix.

▶ **Definition 32** (Success in $\widehat{\lambda}^{\natural}_{\oplus}$). *We extend the syntax of terms for* $\widehat{\lambda}^{\natural}_{\oplus}$ *with the* ✓ *construct. We define* $\mathbb{M} \Downarrow_{\checkmark}$ *iff there exist* $M_1, \cdots, M_k$ *such that* $\mathbb{M} \longrightarrow^{*} M_1 + \cdots + M_k$ *and* $\mathsf{head}(M'_j) = \checkmark$, *for some* $j \in \{1, \ldots, k\}$ *and term* M'_j *such that* $M_j \succeq_{\lambda} M'_j$.

▶ **Definition 33** (Success in $\mathsf{s}\pi$). *We extend the syntax of* $\mathsf{s}\pi$ *processes with the* ✓ *construct, which we assume well typed. We define* $P \Downarrow_{\checkmark}$ *to hold whenever there exists a* P' *such that* $P \longrightarrow^{*} P'$ *and* P' *contains an unguarded occurrence of* ✓.

We now extend Def. 24 by decreeing $[\![\checkmark]\!]^{\natural}_{u} = \checkmark$. We finally have:

▶ **Theorem 34** (Success Sensitivity). *Let* $\mathbb{M}$ *be a well-formed, closed* $\widehat{\lambda}^{\natural}_{\oplus}$ *expression. Then* $\mathbb{M} \Downarrow_{\checkmark}$ *iff* $[\![\mathbb{M}]\!]^{\natural}_{u} \Downarrow_{\checkmark}$.

5 Discussion

Summary. We developed a correct encoding of $\lambda^{\lightning}_{\oplus}$, a new resource λ-calculus in which expressions feature non-determinism and explicit failure, into $\mathsf{s}\pi$, a session-typed π-calculus in which behavior is non-deterministically available: a protocol may perform as stipulated but also fail. Our encodability result is obtained by appealing to $\widehat{\lambda}^{\lightning}_{\oplus}$, an intermediate language with sharing constructs that simplifies the treatment of variables in expressions. To our knowledge, we are the first to relate typed λ-calculi and typed π-calculi encompassing non-determinism and explicit failures, while connecting intersection types and session types, two different mechanisms for resource-awareness in sequential and concurrent settings, respectively.

Design of $\lambda^{\lightning}_{\oplus}$ (and $\widehat{\lambda}^{\lightning}_{\oplus}$). The design of the sequential calculus $\lambda^{\lightning}_{\oplus}$ has been influenced by the typed mechanisms for non-determinism and failure in the concurrent calculus $\mathsf{s}\pi$. As $\mathsf{s}\pi$ stands on rather solid logical foundations (via the Curry-Howard correspondence between linear logic and session types [7, 27, 6]), $\lambda^{\lightning}_{\oplus}$ defines a logically motivated addition to resource λ-calculi in the literature; see, e.g., [3, 4, 21]. Major similarities between $\lambda^{\lightning}_{\oplus}$ and these existing languages include: as in [4], our semantics performs lazy evaluation and linear substitution on the head variable; as in [21], our reductions lead to non-deterministic sums. A distinctive feature of $\lambda^{\lightning}_{\oplus}$ is its lazy treatment of failures, via the dedicated term $\widetilde{\mathtt{fail}^{x}}$. In contrast, in [3, 4, 21] there is no dedicated term to represent failure. The non-collapsing semantics for non-determinism is another distinctive feature of $\lambda^{\lightning}_{\oplus}$.

Our design for $\widehat{\lambda}^{\lightning}_{\oplus}$ has been informed by the λ-calculi with sharing introduced in [13] and studied in [10]. Also, our translation from $\lambda^{\lightning}_{\oplus}$ into $\widehat{\lambda}^{\lightning}_{\oplus}$ borrows insights from the translations presented in [13]. Notice that the calculi in [13, 10] do not consider explicit failure nor non-determinism. We distinguish between *well-typed* and *well-formed* expressions: this allows us to make fail-prone evaluation in $\lambda^{\lightning}_{\oplus}$ explicit. It is interesting that explicit failures can be elegantly encoded as protocols in $\mathsf{s}\pi$– this way, we make the most out of $\mathsf{s}\pi$'s expressivity.

Related Works. A source of inspiration for our work is the work by Boudol and Laneve [4]. As far as we know, this is the only prior study that connects λ and π from a resource-oriented perspective, via an encoding of a λ-calculus with multiplicities into a π-calculus without sums. The goal of [4] is different from ours, as they study the discriminating power of semantics for λ as induced by encodings into π. In contrast, we study how typability delineates the encodability of resource-awareness across sequential and concurrent realms. Notice that the calculi in [4] are untyped, whereas we consider typed calculi and our encodings preserve typability. As a result, the encoding in [4] is conceptually different from ours; remarkably, our encoding of $\widehat{\lambda}^{\lightning}_{\oplus}$ into $\mathsf{s}\pi$ respects linearity and homomorphically translates sums.

There are some similarities between $\lambda^{\lightning}_{\oplus}$ and the differential λ-calculus, introduced in [9]. Both express non-deterministic choice via sums and use linear head reduction for evaluation. In particular, our fetch rule, which consumes non-deterministically elements from a bag, is related to the derivation (which has similarities with substitution) of a differential term. However, the focus of [9] is not on typability nor encodings to process calculi; instead they relate the Taylor series of analysis to the linear head reduction of λ-calculus.

Prior works have studied encodings of typed λ-calculi into typed π-calculi; see, e.g., [23, 4, 24, 1, 16, 20, 26]. None of these works consider non-determinism and failures; the one exception is the encoding in [6], which involves a λ-calculus with exceptions and failures (but without non-determinism due to bags, as in $\lambda^{\lightning}_{\oplus}$) for which no (reduction) semantics is given. As a result, the encoding in [6] is different from ours, and only preserves typability: important semantic properties such as operational completeness, operational soundness, and success sensitivity are not considered in [6].

Ongoing and Future Work. In $\lambda^{\lightning}_{\oplus}$ bags have *linear* resources, which are used exactly once. In ongoing work, we have established that our approach to encodability in $\mathsf{s}\pi$ extends to the case in which bags contain both linear and *unrestricted* resources, as in [21]. Handling such an extension of $\lambda^{\lightning}_{\oplus}$ requires the full typed process framework in [6], with replicated processes and labeled choices (which were not needed to encode $\lambda^{\lightning}_{\oplus}$).

The approach and results developed here enable us to tackle open questions that go beyond the scope of this work. First, we wish to explore whether our correct encoding can be defined in a setting with *collapsing* non-determinism. Second, we plan to investigate formal results of relative expressiveness that connect $\lambda^{\lightning}_{\oplus}$ and the resource calculi in [4, 21].

References

1 Martin Berger, Kohei Honda, and Nobuko Yoshida. 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*, volume 2620 of *Lecture Notes in Computer Science*, pages 103–119. Springer, 2003. doi:10.1007/3-540-36576-1_7.

2 Viviana Bono and Mariangiola Dezani-Ciancaglini. A tale of intersection types. In Holger Hermanns, Lijun Zhang, Naoki Kobayashi, and Dale Miller, editors, *LICS '20: 35th Annual ACM/IEEE Symposium on Logic in Computer Science, Saarbrücken, Germany, July 8-11, 2020*, pages 7–20. ACM, 2020. doi:10.1145/3373718.3394733.

3 Gérard Boudol. The lambda-calculus with multiplicities (abstract). In Eike Best, editor, *CONCUR '93, Hildesheim, Germany, August 23-26, 1993, Proceedings*, volume 715 of *Lecture Notes in Computer Science*, pages 1–6. Springer, 1993. doi:10.1007/3-540-57208-2_1.

4 Gérard Boudol and Cosimo Laneve. lambda-calculus, multiplicities, and the pi-calculus. In *Proof, Language, and Interaction, Essays in Honour of Robin Milner*, pages 659–690, 2000.

5 Antonio Bucciarelli, Delia Kesner, and Daniel Ventura. Non-idempotent intersection types for the lambda-calculus. *Logic Journal of the IGPL*, 25(4):431–464, 2017.

6 Luís Caires and Jorge A. Pérez. 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*, volume 10201 of *Lecture Notes in Computer Science*, pages 229–259. Springer, 2017. doi:10.1007/978-3-662-54434-1_9.

7 Luís Caires and Frank Pfenning. Session types as intuitionistic linear propositions. In *CONCUR 2010 – Concurrency Theory, 21th International Conference, CONCUR 2010, Paris, France, August 31–September 3, 2010. Proceedings*, pages 222–236, 2010. doi:10.1007/978-3-642-15375-4_16.

8 Mariangiola Dezani-Ciancaglini, Ugo de'Liguoro, and Adolfo Piperno. Filter models for a parallel and non deterministic lambda-calculus. In Andrzej M. Borzyszkowski and Stefan Sokolowski, editors, *Mathematical Foundations of Computer Science 1993, 18th International Symposium, MFCS'93, Gdansk, Poland, August 30–September 3, 1993, Proceedings*, volume 711 of *Lecture Notes in Computer Science*, pages 403–412. Springer, 1993. doi:10.1007/3-540-57182-5_32.

9 Thomas Ehrhard and Laurent Regnier. The differential lambda-calculus. *Theor. Comput. Sci.*, 309(1-3):1–41, 2003. doi:10.1016/S0304-3975(03)00392-X.

10 Silvia Ghilezan, Jelena Ivetic, Pierre Lescanne, and Silvia Likavec. 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*, pages 116–134, 2011. doi:10.1007/978-3-642-23283-1_10.

11 Daniele Gorla. Towards a unified approach to encodability and separation results for process calculi. *Inf. Comput.*, 208(9):1031–1053, 2010. doi:10.1016/j.ic.2010.05.002.

12 Daniele Gorla and Uwe Nestmann. Full abstraction for expressiveness: history, myths and facts. *Math. Struct. Comput. Sci.*, 26(4):639–654, 2016. doi:10.1017/S0960129514000279.

13 Tom Gundersen, Willem Heijltjes, and Michel Parigot. 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*, pages 311–320, 2013. doi:10.1109/LICS.2013.37.

14 Kohei Honda. Types for dyadic interaction. In Eike Best, editor, *CONCUR '93, Hildesheim, Germany, August 23-26, 1993, Proceedings*, volume 715 of *Lecture Notes in Computer Science*, pages 509–523. Springer, 1993. doi:10.1007/3-540-57208-2_35.

15 Kohei Honda, Vasco Thudichum Vasconcelos, and Makoto Kubo. 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*, volume 1381 of *Lecture Notes in Computer Science*, pages 122–138. Springer, 1998. doi:10.1007/BFb0053567.

16 Kohei Honda, Nobuko Yoshida, and Martin Berger. 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*, volume 8560 of *Lecture Notes in Computer Science*, pages 1–20. Springer, 2014. doi:10.1007/978-3-319-08918-8_1.

17 Dimitrios Kouzapas, Jorge A. Pérez, and Nobuko Yoshida. On the relative expressiveness of higher-order session processes. *Inf. Comput.*, 268, 2019. doi:10.1016/j.ic.2019.06.002.

18 Robin Milner. Functions as processes. *Mathematical Structures in Computer Science*, 2(2):119–141, 1992. doi:10.1017/S0960129500001407.

19 Robin Milner, Joachim Parrow, and David Walker. A calculus of mobile processes, I. *Inf. Comput.*, 100(1):1–40, 1992. doi:10.1016/0890-5401(92)90008-4.

20 Dominic A. Orchard and Nobuko Yoshida. Effects as sessions, sessions as effects. In *Proceedings of the 43rd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2016, St. Petersburg, FL, USA, January 20–22, 2016*, pages 568–581. ACM, 2016. doi:10.1145/2837614.2837634.

21 Michele Pagani and Simona Ronchi Della Rocca. 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*, volume 6014 of *Lecture Notes in Computer Science*, pages 358–373. Springer, 2010. doi:10.1007/978-3-642-12032-9_25.

22 Joseph Paulus, Daniele Nantes-Sobrinho, and Jorge A. Pérez. Non-Deterministic Functions as Non-Deterministic Processes (Extended Version). *CoRR*, abs/2104.14759, 2021. arXiv:2104.14759.

23 Davide Sangiorgi. From lambda to pi; or, rediscovering continuations. *Math. Struct. Comput. Sci.*, 9(4):367–401, 1999. URL: http://journals.cambridge.org/action/displayAbstract?aid=44843.

24 Davide Sangiorgi and David Walker. *The Pi-Calculus – a theory of mobile processes*. Cambridge University Press, 2001.

25 Bernardo Toninho, Luís Caires, and Frank Pfenning. 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*, volume 7213 of *Lecture Notes in Computer Science*, pages 346–360. Springer, 2012. doi:10.1007/978-3-642-28729-9_23.

26 Bernardo Toninho and Nobuko Yoshida. 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*, volume 10801 of *Lecture Notes in Computer Science*, pages 827–855. Springer, 2018. doi:10.1007/978-3-319-89884-1_29.

27 Philip Wadler. Propositions as sessions. In Peter Thiemann and Robby Bruce Findler, editors, *ACM SIGPLAN International Conference on Functional Programming, ICFP'12, Copenhagen, Denmark, September 9-15, 2012*, pages 273–286. ACM, 2012. doi:10.1145/2364527.2364568.

A Appendix

A.1 Omitted Syntactic and Semantic Notations for $\lambda^{\lightning}_{\oplus}$

Auxiliary Notions. In $\lambda^{\lightning}_{\oplus}$, a β-reduction induces an explicit substitution of a bag B for a variable x, denoted $\langle\!\langle B/x \rangle\!\rangle$. This explicit substitution is then expanded into a sum of terms, each of which features a *linear head substitution* $\{\!|N_i/x|\!\}$, where N_i is a term in B; the bag $B \setminus N_i$ is kept in an explicit substitution. In case there is a mismatch between the number of occurrences of the variable to be substituted and the number of resources available, then the reduction leads to the failure term. The reduction rules in Fig. 12 rest upon some auxiliary notions.

▶ **Definition 35** (Set and Multiset of Free Variables). *The set of free variables of a term, bag, and expression, is defined in Fig. 11. We use* $\mathsf{mfv}(M)$ *or* $\mathsf{mfv}(B)$ *to denote a multiset of free variables, defined similarly. We sometimes treat the sequence* $\widetilde{x}$ *as a (multi)set. We write* $\widetilde{x} \uplus \widetilde{y}$ *to denote the multiset union of* $\widetilde{x}$ *and* $\widetilde{y}$ *and* $\widetilde{x} \setminus y$ *to express that every occurrence of* y *is removed from* $\widetilde{x}$*. As usual, a term* M *is* closed *if* $\mathsf{fv}(M) = \emptyset$ *(and similarly for expressions).*

$$\mathsf{fv}(x) = \{x\} \quad \mathsf{fv}(\wr M \wr) = \mathsf{fv}(M) \quad \mathsf{fv}(\lambda x.M) = \mathsf{fv}(M) \setminus \{x\} \quad \mathsf{fv}(M\ B) = \mathsf{fv}(M) \cup \mathsf{fv}(B)$$
$$\mathsf{fv}(1) = \emptyset \quad \mathsf{fv}(B_1 \cdot B_2) = \mathsf{fv}(B_1) \cup \mathsf{fv}(B_2) \quad \mathsf{fv}(\mathbb{M} + \mathbb{N}) = \mathsf{fv}(\mathbb{M}) \cup \mathsf{fv}(\mathbb{N})$$
$$\mathsf{fv}(M\langle\!\langle B/x \rangle\!\rangle) = (\mathsf{fv}(M) \setminus \{x\}) \cup \mathsf{fv}(B) \quad \mathsf{fv}(\mathtt{fail}^{x_1,\cdots,x_n}) = \{x_1, \cdots, x_n\}$$

Figure 11 Free variables for $\lambda^{\lightning}_{\oplus}$.

▶ **Notation 36.** *$\#(x, M)$ denotes the number of (free) occurrences of x in M. Similarly, we write $\#(x, \widetilde{y})$ to denote the number of occurrences of x in the multiset $\widetilde{y}$.*

▶ **Definition 37** (Head). *Given a term M, we define* $\mathsf{head}(M)$ *inductively as:*

$$\mathsf{head}(x) = x \quad \mathsf{head}(\lambda x.M) = \lambda x.M \quad \mathsf{head}(M\ B) = \mathsf{head}(M)$$
$$\mathsf{head}(\mathtt{fail}^{\widetilde{x}}) = \mathtt{fail}^{\widetilde{x}}$$
$$\mathsf{head}(M\langle\!\langle B/x \rangle\!\rangle) = \begin{cases} \mathsf{head}(M) & \text{if } \#(x, M) = \mathsf{size}(B) \\ \mathtt{fail}^{\emptyset} & \text{otherwise} \end{cases}$$

▶ **Definition 38** (Linear Head Substitution). *Let M be a term such that* $\mathsf{head}(M) = x$*. The* linear head substitution *of a term N for x, denoted $\{\!|N/x|\!\}$, is defined as:*

$$x\{\!|N/x|\!\} = N \quad (M\ B)\{\!|N/x|\!\} = (M\{\!|N/x|\!\})\ B$$
$$(M\ \langle\!\langle B/y \rangle\!\rangle)\{\!|N/x|\!\} = (M\{\!|N/x|\!\})\ \langle\!\langle B/y \rangle\!\rangle \quad \text{where } x \neq y$$

$$[\mathtt{R}:\mathtt{Beta}]\ \frac{}{(\lambda x.M)B \longrightarrow M\ \langle\!\langle B/x \rangle\!\rangle} \qquad [\mathtt{R}:\mathtt{Fail}]\ \frac{\#(x,M) \neq \mathsf{size}(B) \quad \widetilde{y} = (\mathsf{mfv}(M) \setminus x) \uplus \mathsf{mfv}(B)}{M\ \langle\!\langle B/x \rangle\!\rangle \longrightarrow \sum_{\mathsf{PER}(B)} \mathtt{fail}^{\widetilde{y}}}$$

$$[\mathtt{R}:\mathtt{Fetch}]\ \frac{\mathsf{head}(M) = x \qquad B = \wr N_1 \wr \cdot \cdots \cdot \wr N_k \wr\ ,\ k \geq 1 \qquad \#(x,M) = k}{M\ \langle\!\langle B/x \rangle\!\rangle \longrightarrow M\{\!|N_1/x|\!\}\langle\!\langle (B \setminus N_1)/x \rangle\!\rangle + \cdots + M\{\!|N_k/x|\!\}\langle\!\langle (B \setminus N_k)/x \rangle\!\rangle}$$

$$[\mathtt{R}:\mathtt{Cons}_1]\ \frac{\widetilde{y} = \mathsf{mfv}(B)}{\mathtt{fail}^{\widetilde{x}}\ B \longrightarrow \sum_{\mathsf{PER}(B)} \mathtt{fail}^{\widetilde{x \uplus y}}} \qquad [\mathtt{R}:\mathtt{Cons}_2]\ \frac{\mathsf{size}(B) = k \qquad \#(z,\widetilde{x}) + k \neq 0 \quad \widetilde{y} = \mathsf{mfv}(B)}{\mathtt{fail}^{\widetilde{x}}\ \langle\!\langle B/z \rangle\!\rangle \longrightarrow \sum_{\mathsf{PER}(B)} \mathtt{fail}^{\widetilde{(x \setminus z) \uplus y}}}$$

$$[\mathtt{R}:\mathtt{TCont}]\ \frac{M \longrightarrow M'_1 + \cdots + M'_k}{C[M] \longrightarrow C[M'_1] + \cdots + C[M'_k]} \qquad [\mathtt{R}:\mathtt{ECont}]\ \frac{\mathbb{M} \longrightarrow \mathbb{M}'}{D[\mathbb{M}] \longrightarrow D[\mathbb{M}']}$$

Figure 12 Reduction rules for $\lambda^{\wr}_{\oplus}$.

Finally, we define contexts for terms and expressions and convenient notations:

▶ **Definition 39** (Term and Expression Contexts). *Contexts for terms (CTerm) and expressions (CExpr) are defined by the following grammar:*

$$\textit{(CTerm)}\ C[\cdot], C'[\cdot] ::= ([\cdot])B \mid ([\cdot])\langle\!\langle B/x \rangle\!\rangle \qquad \textit{(CExpr)}\ D[\cdot], D'[\cdot] ::= M + [\cdot] \mid [\cdot] + M$$

Reduction for $\boldsymbol{\lambda^{\wr}_{\oplus}}$. The reduction relation $\longrightarrow$ operates lazily on expressions; it is defined by the rules in Fig. 12. Rule $[\mathtt{R}:\mathtt{Beta}]$ is standard and admits a bag (possibly empty) as parameter. Rule $[\mathtt{R}:\mathtt{Fetch}]$ transforms a term into an expression: it opens up an explicit substitution into a sum of terms with linear head substitutions, each denoting the partial evaluation of an element from the bag. Hence, the size of the bag will determine the number of summands in the resulting expression.

Three rules reduce to the failure term: their objective is to accumulate all (free) variables involved in failed reductions. Accordingly, Rule $[\mathtt{R}:\mathtt{Fail}]$ formalizes failure in the evaluation of an explicit substitution $M\ \langle\!\langle B/x \rangle\!\rangle$, which occurs if there is a mismatch between the resources (terms) present in B and the number of occurrences of x to be substituted. The resulting failure preserves all free variables in M and B within its attached multiset $\widetilde{y}$. Rules $[\mathtt{R}:\mathtt{Cons}_1]$ and $[\mathtt{R}:\mathtt{Cons}_2]$ describe reductions that lazily consume the failure term, when a term has $\mathtt{fail}^{\widetilde{x}}$ at its head position. The former rule consumes bags attached to it whilst preserving all its free variables. The latter rule is similar but for the case of explicit substitutions; its second premise ensures that either (i) the bag in the substitution is not empty or (ii) the number of occurrences of x in the current multiset of accumulated variables is not zero. When both (i) and (ii) hold, we apply a precongruence rule (cf. [22]), rather than reduction.

Finally, Rule $[\mathtt{R}:\mathtt{TCont}]$ describes the reduction of sub-terms within an expression; in this rule, summations are expanded outside of term contexts. Rule $[\mathtt{R}:\mathtt{ECont}]$ says that reduction of expressions is closed by expression contexts.

▶ **Example 40.** Let $M = (\lambda x.x\{\!|x\{\!|y|\!\}|\!\})\ B$, with $B = \{\!|z_1|\!\} \cdot \{\!|z_2|\!\} \cdot \{\!|z_1|\!\}$. We have:

$$M \longrightarrow_{[\texttt{R:Beta}]} x\{\!|x\{\!|y|\!\}|\!\}\langle\!\langle \{\!|z_1|\!\} \cdot \{\!|z_2|\!\} \cdot \{\!|z_1|\!\} / x \rangle\!\rangle \longrightarrow_{[\texttt{R:Fail}]} \sum_{\mathsf{PER}(B)} \texttt{fail}^{y,z_1,z_2,z_1}$$

The number of occurrences of x in the term obtained after β-reduction (2) does not match the size of the bag (3). Therefore, the reduction leads to failure.

Notice that the left-hand sides of the reduction rules in $\lambda^{\not\{}_{\oplus}$ do not interfere with each other. Therefore, reduction in $\lambda^{\not\{}_{\oplus}$ satisfies a *diamond property*:

▶ **Proposition 41** (Diamond Property for $\lambda^{\not\{}_{\oplus}$)**.** *For all* $\mathbb{N}$, $\mathbb{N}_1$, $\mathbb{N}_2$ *in* $\lambda^{\not\{}_{\oplus}$ *s.t.* $\mathbb{N} \longrightarrow \mathbb{N}_1$, $\mathbb{N} \longrightarrow \mathbb{N}_2$ *with* $\mathbb{N}_1 \neq \mathbb{N}_2$ *then* $\exists \mathbb{M}$ *s.t.* $\mathbb{N}_1 \longrightarrow \mathbb{M}$, $\mathbb{N}_2 \longrightarrow \mathbb{M}$.

Proof. We give a short argument to convince the reader of this. Notice that an expression can only perform a choice of reduction steps when it is a nondeterministic sum of terms in which multiple terms can perform independent reductions. For simplicity sake we will only consider an expression $\mathbb{N}$ that consist of two terms where $\mathbb{N} = N + M$. We also have that $N \longrightarrow N'$ and $M \longrightarrow M'$. Then we let $\mathbb{N}_1 = N' + M$ and $\mathbb{N}_2 = N + M'$ by the [$\texttt{R} : \texttt{ECont}$] rules. Finally we prove that $\mathbb{M}$ exists by letting $\mathbb{M} = N' + M'$ ◀

Non-Idempotent Intersection Types. The type system for $\lambda^{\not\{}_{\oplus}$ is based on non-idempotent intersection types. The grammar of strict and multiset types, the notions of typing assignments and judgements are the same as in Section 2.

We define *well-formed* $\lambda^{\not\{}_{\oplus}$ expressions, in two stages. We first define a type system for the sub-language $\lambda_{\oplus}$, given in Fig. 13, using the types of Def. 12. Then, we define well-formed expressions for the full language $\lambda^{\not\{}_{\oplus}$, via Def. 42 (see below).

We first discuss selected rules of the type system for $\lambda_{\oplus}$ in Fig. 13. Rule [T:var] is standard. Rule [T:1] assigns the empty bag 1 the empty type ω. Rule [T:weak] introduces a useful weakening principle. Rule [T:app] is standard, requiring a match on the multiset type π. Rule [T:ex-sub] types explicit substitutions where a bag must consist of both the same type and size of the variable it is being substituted for. On top of this type system for $\lambda_{\oplus}$, we define well-formed expressions:

▶ **Definition 42** (Well-formed $\lambda^{\not\{}_{\oplus}$ expressions)**.** *An expression* $\mathbb{M}$ *is well-formed if there exist* Γ *and* τ *such that* $\Gamma \models \mathbb{M} : \tau$ *is entailed via the rules in Fig. 14.*

In Fig. 14, Rules [F:wf-expr] and [F:wf-bag] allow well-typed terms and bags to be well-formed. Rules [F:abs], [F:bag], and [F:sum] are as in the type system for $\lambda_{\oplus}$, but extended to the system of well-formed expressions. Rules [F:ex-sub] and [F:app] differ from similar typing rules as the size of the bags (as declared in their types) is no longer required to match. Finally, Rule [F:fail] has no analogue in the type system: we allow the failure term $\texttt{fail}^{\widetilde{x}}$ to be well-formed with any type, provided that the context contains the types of the variables in $\widetilde{x}$.

Well-formed expressions satisfy subject reduction (SR); see [22] for a proof.

▶ **Theorem 43** (SR in $\lambda^{\not\{}_{\oplus}$)**.** *If* $\Gamma \models \mathbb{M} : \tau$ *and* $\mathbb{M} \longrightarrow \mathbb{M}'$ *then* $\Gamma \models \mathbb{M}' : \tau$.

Clearly, the set of *well-typed* expressions is strictly included in the set of *well-formed* expressions. Take $M = x\langle\!\langle \{\!|N_1|\!\} \cdot \{\!|N_2|\!\} / x \rangle\!\rangle$ where both N_1 and N_2 are well-typed. It is easy to see that M is well-formed. However, M is not well-typed.

▶ **Example 44.** The following example illustrates an expression which is not well-formed:

$$\lambda x.x\{\!|\lambda y.y|\!\} \cdot \{\!|\lambda z.z_1\{\!|z_1\{\!|z_2|\!\}|\!\}|\!\}$$

This is due to the bag being composed of two terms of different types.

$$[\mathtt{T}:\mathtt{var}]\ \frac{}{x:\sigma \vdash x:\sigma} \qquad [\mathtt{T}:\mathtt{1}]\ \frac{}{\vdash \mathtt{1}:\omega} \qquad [\mathtt{T}:\mathtt{weak}]\ \frac{\Gamma \vdash M:\sigma \qquad x \notin \mathtt{dom}(\Gamma)}{\Gamma, x:\omega \vdash M:\sigma}$$

$$[\mathtt{T}:\mathtt{abs}]\ \frac{\Gamma, \hat{x}:\sigma^k \vdash M:\tau \quad x \notin \mathtt{dom}(\Gamma)}{\Gamma \vdash \lambda x.M:\sigma^k \to \tau} \qquad [\mathtt{T}:\mathtt{app}]\ \frac{\Gamma \vdash M:\pi \to \tau \qquad \Delta \vdash B:\pi}{\Gamma, \Delta \vdash M\ B:\tau}$$

$$[\mathtt{T}:\mathtt{bag}]\ \frac{\Gamma \vdash M:\sigma \qquad \Delta \vdash B:\sigma^k}{\Gamma, \Delta \vdash \{\!|M|\!\} \cdot B:\sigma^{k+1}} \qquad [\mathtt{T}:\mathtt{sum}]\ \frac{\Gamma \vdash \mathbb{M}:\sigma \qquad \Gamma \vdash \mathbb{N}:\sigma}{\Gamma \vdash \mathbb{M}+\mathbb{N}:\sigma}$$

$$[\mathtt{T}:\mathtt{ex\text{-}sub}]\ \frac{\Gamma, \hat{x}:\sigma^k \vdash M:\tau \qquad \Delta \vdash B:\sigma^k}{\Gamma, \Delta \vdash M\langle\!\langle B/x \rangle\!\rangle:\tau}$$

Figure 13 Typing rules for the sub-language $\lambda_{\oplus}$ (i.e., $\lambda^{\not\iota}_{\oplus}$ without the failure term).

$$[\mathtt{F}:\mathtt{wf\text{-}expr}]\ \frac{\Gamma \vdash \mathbb{M}:\tau}{\Gamma \models \mathbb{M}:\tau} \qquad [\mathtt{F}:\mathtt{wf\text{-}bag}]\ \frac{\Gamma \vdash B:\pi}{\Gamma \models B:\pi} \qquad [\mathtt{F}:\mathtt{weak}]\ \frac{\Delta \models M:\tau}{\Delta, x:\omega \models M:\tau}$$

$$[\mathtt{F}:\mathtt{abs}]\ \frac{\Gamma, \hat{x}:\sigma^n \models M:\tau \quad x \notin \mathtt{dom}(\Gamma)}{\Gamma \models \lambda x.M:\sigma^n \to \tau} \qquad [\mathtt{F}:\mathtt{bag}]\ \frac{\Gamma \models M:\sigma \qquad \Delta \models B:\sigma^k}{\Gamma, \Delta \models \{\!|M|\!\} \cdot B:\sigma^{k+1}}$$

$$[\mathtt{F}:\mathtt{sum}]\ \frac{\Gamma \models \mathbb{M}:\sigma \qquad \Gamma \models \mathbb{N}:\sigma}{\Gamma \models \mathbb{M}+\mathbb{N}:\sigma} \qquad [\mathtt{F}:\mathtt{fail}]\ \frac{\mathtt{dom}(\Gamma) = \widetilde{x}}{\Gamma \models \mathtt{fail}^{\widetilde{x}}:\tau}$$

$$[\mathtt{F}:\mathtt{ex\text{-}sub}]\ \frac{\Gamma, \hat{x}:\sigma^k \models M:\tau \qquad \Delta \models B:\sigma^j \quad k,j \geq 0}{\Gamma, \Delta \models M\langle\!\langle B/x \rangle\!\rangle:\tau}$$

$$[\mathtt{F}:\mathtt{app}]\ \frac{\Gamma \models M:\sigma^j \to \tau \qquad \Delta \models B:\sigma^k \qquad k,j \geq 0}{\Gamma, \Delta \models M\ B:\tau}$$

Figure 14 Well-formedness rules for the full language $\lambda^{\not\iota}_{\oplus}$.