Formalizing Functions as Processes

Beniamino Accattoli, Horace Blanc, Claudio Sacerdoti Coen

Formalizing Functions as Processes

Beniamino Accattoli
Inria & LIX, École Poytechnique, Palaiseau, France

Horace Blanc
Independent researcher, Paris, France

Claudio Sacerdoti Coen
Alma Mater Studiorum - University of Bologna, Italy

Abstract

We present the first formalization of Milner's classic translation of the λ-calculus into the π-calculus. It is a challenging result with respect to variables, names, and binders, as it requires one to relate variables and binders of the λ-calculus with names and binders in the π-calculus. We formalize it in Abella, merging the set of variables and the set of names, thus circumventing the challenge and obtaining a neat formalization.

About the translation, we follow Accattoli's factoring of Milner's result via the linear substitution calculus, which is a λ-calculus with explicit substitutions and contextual rewriting rules, mediating between the λ-calculus and the π-calculus. Another aim of the formalization is to investigate to which extent the use of contexts in Accattoli's refinement can be formalized.

2012 ACM Subject Classification Theory of computation → Lambda calculus; Theory of computation → Process calculi; Theory of computation → Operational semantics; Theory of computation → Automated reasoning

Keywords and phrases Lambda calculus, pi calculus, proof assistants, binders, Abella

Digital Object Identifier 10.4230/LIPIcs.ITP.2023.5

Supplementary Material *Software (Abella sources)*: `github.com/sacerdot/FunctionsAsProcesses` archived at `swh:1:dir:3237e6680f66745d679f6976470e9a7bb465c931`

Funding Sacerdoti Coen has been supported by the INdAM-GNCS project "Proprietà qualitative e quantitative dei Sistemi Reversibili" and by the Cost Action CA20111 EuroProofNet.

1 Introduction

Milner's translation of the λ-calculus in the π-calculus [32] is a classic result relating two paradigmatic formalisms. It gave rise to many studies, most notably by Sangiorgi, both alone and with co-authors [39, 40, 18, 42, 27, 20], but also *e.g.* by Boudol [15, 14], Niehren [35], Kobayashi [28], Cai and Fu [16], Toninho et al. [44], and Biernacka et al. [13].

Properties of the π-calculus have been formalized a number of times, for instance by Melham [29], Aït Mohamed [34], Hirschkoff [25], Despeyroux [19], Röckl et al. [38], Honsell et al. [26], Gay [24], Bengston and Parrow [12], Gabbay [21], Chauduri et al. [17], Orchard and Yoshida [36], Perera and Cheney [37], Veltri and Vezzosi [45], and Ambal et al. [10]. To our knowledge, however, the correctness of the translation of λ into π has never been formalized. Miller and Nadathur implement the translation in [30] (p. 274), but not the proof of correctness. In [25], Hirschkoff clearly states that his work is preliminary to the formalization of the correctness of the translation. He also says that one of the main obstacles is the correspondence between term variables and process names, and that *"some work should be done to reformulate some parts of the proof in a way that would be more tractable for the task of mechanisation"*. This was in 1997. In the meantime, the theory of proof assistants has developed a variety of tools for dealing with names and binders, and Milner's translation has indeed been refactored, as we discuss below.

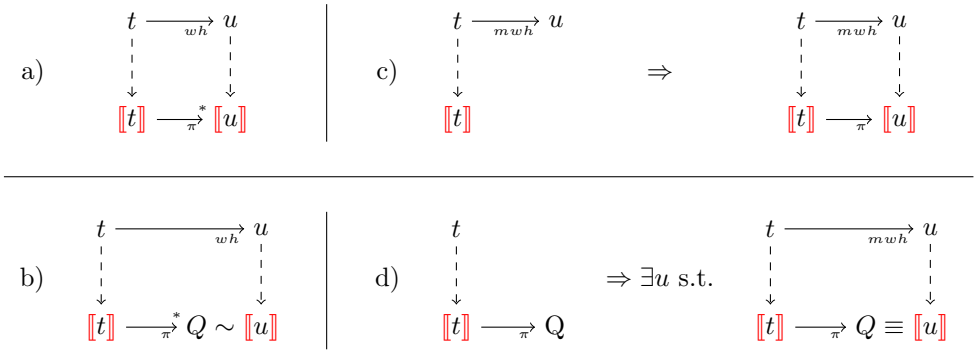

Figure 1 Diagrams describing the relationships between terms and processes.

Variable, Names, and Binders. The translation of λ into π poses additional difficulties with respect to studying a single language with bindings:

1. *Variables/names relation*: one has to relate two languages with binders and establish a relationship between variables and binders in a term t and the names and binders in its representation $[\![t]\!]$ as a process, in order to prove that rewriting steps on both sides preserve the translation;
2. *Not a bijection, and issues with de Bruijn*: the process $[\![t]\!]$ uses more binders than t. Typically, applications (which have no binders) are represented in π via the addition of various binders. Therefore, adopting de Bruijn representations of terms and processes, a variable occurrence of index $i \in \mathbb{N}$ in t is not represented with the same index in $[\![t]\!]$.
3. *Structural equivalence*: reduction on processes is defined only *up to* structural equivalence, which re-organizes the structure of binders by moving restrictions operators around.

Small-Step *vs* Micro-Step. Names are not the only difficulty. The ways in which the λ-calculus and the π-calculus compute are inherently different. A first aspect is that the π-calculus does not compute under prefixes, which corresponds to *weak* evaluation on the λ-calculus, that is, to not compute under abstractions. This is however not the key point.

The λ-calculus rests on a *small-step* operational semantics, based on meta-level substitution, which replaces all occurrences of a variable at the same time with a whole sub-term. The π-calculus instead has a *micro-step* approach: it only substitutes names, not sub-processes, and does the analogous of replacing only one occurrence at a time, keeping sort of *explicit substitutions* for the names that have not been fully substituted yet. In the context of λ-calculi, the relationship between small-step and micro-step evaluation is well studied, and dealt with via the notion of *unfolding* of explicit substitutions, which turns explicit substitutions into meta-level ones. The problem, in the case of the π-calculus, is that the unfolding operation has no natural analogous on processes.

The small-*vs*-micro-step issue implies that the simulation of λ into π, which relates *weak head reduction* $\to_{wh}$ on the λ-calculus with reduction in π, is not as strong as one might expect. The diagram in Fig. 1.a indeed does *not* hold. One only has the diagram in Fig. 1.b, for which $[\![t]\!]$ reduces to a process Q which is *strongly bisimilar* to $[\![u]\!]$, that is, that behaves equivalently *externally* (*i.e.* with respect to the environment), but which is in general very different from $[\![u]\!]$ both structurally and with respect to *internal* reductions.

Refactoring of the Translation. In his seminal work, Milner shows that Q can be seen as the representation of a term $[\![r]\!]$ plus a set of processes corresponding to explicit substitutions $[x_1 \leftarrow [\![w_1]\!]] \dots [x_k \leftarrow [\![w_k]\!]]$, and that $[\![u]\!] = [\![r]\!]\{x_1 \leftarrow [\![w_1]\!]\} \dots \{x_k \leftarrow [\![w_k]\!]\}$, that is, turning the explicit substitutions into meta-level ones and applying them to $[\![r]\!]$ (which is the unfolding of Q, if seen as a λ-term with explicit substitutions) gives $[\![u]\!]$.

In 2013, Accattoli refines Milner's argument factoring the translation of λ into π via a simple λ-calculus with explicit substitutions [2], namely Accattoli and Kesner's *linear substitution calculus* (LSC). The LSC is itself a refinement of a calculus by Milner [33], and can be considered as the canonical micro-step λ-calculus: Accattoli, Kesner, and co-authors have shown that it is strongly related to abstract machines [4, 8], linear logic proof nets [3], reasonable cost models [7], multi types [9], and rewriting theory [5]. Back to the translation, in [2] it is shown that the π-calculus evaluates λ-terms *exactly* as the LSC version $\to_{mwh}$ of the weak head strategy $\to_{wh}$ of the λ-calculus. Once one replaces $\to_{wh}$ with $\to_{mwh}$, indeed, the diagrams in Fig. 1.c and Fig. 1.d hold, that is, there is a *bijection* of steps between $\to_{mwh}$ and π, and the translation becomes a *strong bisimulation* between the *internal reductions* of the two (up to structural equivalence $\equiv$), considerably tightening Milner's result. Intuitively, the problem with the unfolding is avoided altogether, since both the LSC and π are micro-step.

The other half of the simulation, that is, the relationship between the LSC and the λ-calculus was studied independently by Accattoli and Dal Lago [6], as part of a study of time cost models for the λ-calculus. It amounts to relate small-step and micro-step notions of substitutions. The factorization via the LSC recasts such a study within a single formalism, since the LSC is an extension of the λ-calculus, and allows one to present it using the unfolding, which is natural in this setting, and makes the study conceptually cleaner.

This Paper. Here we present a formalization of the translation of λ into π, the first one in the literature. We develop it in the Abella proof assistant [22, 11], the first version of which appeared in 2008. It is based on Miller and Nadathur's λ-*tree syntax* [30], a variant of higher-order abstract syntax providing primitive support for binders and capture-avoiding substitution, and Miller and Tiu's ∇ *(nabla) quantifier* [31, 23], providing primitive support for free variables. The difficulties of the translation related to names are fully circumvented by defining terms and processes on the same type, as to share the set of variables/names. And we follow Accattoli and decompose Milner's result in two parts: first, the relationship between the LSC and π, and then the one between the LSC and λ. The resulting formalization is neat and compact.

Contexts and Distance. The LSC is a framework that relies on *contexts*, that is, terms with a hole, and contextual rewriting rules, also called *at a distance*. Beyond the formalization of the translation, this work also explores how to formalize the kind of context-based reasoning arising in the theory of the LSC.

In [2], Accattoli gives a novel contextual presentation of the π-calculus communication rules, the key property of which is that structural equivalence can be postponed. He uses this fact to simplify the proof of the LSC/π correspondence. We also formalize the equivalence of the new presentation with respect to the ordinary one.

2 λ-Terms, Contexts, Rules, and Processes, with Pen and Paper

Here we give the pen-and-paper definitions of the languages involved in our formalization.

λ-Terms and Contexts. We start by defining λ-terms and contexts.

$$\lambda\text{-TERMS}\quad t,u,r \;::=\; x \mid \lambda x.t \mid tu \qquad \text{CONTEXTS}\quad C,D \;::=\; \langle\cdot\rangle \mid \lambda x.C \mid Cu \mid tC$$

Contexts are terms with exactly one hole $\langle\cdot\rangle$, which is a placeholder for a removed sub-term. The basic operation about contexts is *plugging* $C\langle t\rangle$ of a term t for the hole of C, which amounts to simply replacing the hole with t. The tricky aspect of plugging is that it is *not* a capture-avoiding operation, and that it can capture many variables at once, that is, for instance $(\lambda x.\lambda y.\langle\cdot\rangle)\langle xyz\rangle = \lambda x.\lambda y.xyz$. Capture-avoiding plugging is instead denoted with $C\langle\!\langle t\rangle\!\rangle$. While Abella offers primitive first-class support for both binders and free variables, it does not provide support for general contexts. More precisely, it supports contexts with capture-*avoiding* plugging (*i.e.* $C\langle\!\langle t\rangle\!\rangle$), since it is an instance of capture-avoiding substitution (where the bound variable has only one occurrence). What is not available is the capture-allowing operation of plugging (*i.e.* $C\langle t\rangle$). In our development, luckily, in the only point where we really need contexts, we can limit ourselves to a capture-*avoiding* notion of plugging.

Weak Head Reduction. We shall only use the simplest possible evaluation strategy on λ-terms, *weak head reduction* $\to_{wh}$, also known as (weak) *call-by-name* reduction.

$$\text{WEAK HEAD REDUCTION}\quad \frac{}{(\lambda x.t)u \to_{wh} t\{x\leftarrow u\}} \qquad \frac{t \to_{wh} u}{tr \to_{wh} ur}$$

We call *monolithic* the given definition of $\to_{wh}$. It can equivalently be defined in a *split* way, by separating the root rule $\mapsto_{wh}$ and the general rule $\to_{wh}$ obtained by the inductive closure of $\mapsto_{wh}$; or *contextually*, by further compacting the inductive cases via a notion of context.

SPLIT DEFINITION	CONTEXTUAL DEFINITION
$\dfrac{}{(\lambda x.t)u \mapsto_{wh} t\{x\leftarrow u\}}$	APPLICATIVE CONTEXTS $\quad A ::= \langle\cdot\rangle \mid At$
$\dfrac{t \mapsto_{wh} u}{t \to_{wh} u} \qquad \dfrac{t \to_{wh} u}{tr \to_{wh} ur}$	$\dfrac{}{(\lambda x.t)u \mapsto_{wh} t\{x\leftarrow u\}} \qquad \dfrac{t \mapsto_{wh} u}{A\langle t\rangle \to_{wh} A\langle u\rangle}$

These three definitions lead to different formalizations in Abella. For $\to_{wh}$, for which we do not need to prove many properties, we shall adopt the monolithic one. The only two properties of $\to_{wh}$ that we shall use are the following ones.

▶ **Lemma 1.** **1.** Determinism of $\to_{wh}$: *if* $t \to_{wh} u$ *and* $t \to_{wh} r$ *then* $u = r$.
2. Stability under substitution of $\to_{wh}$: *if* $t \to_{wh} u$ *then* $t\{x\leftarrow r\} \to_{wh} u\{x\leftarrow r\}$.

Micro Weak Head Reduction. We do not define the whole reduction of the linear substitution calculus, but only *micro weak head reduction* (also known as *linear* weak head reduction), the micro-step variant of weak head reduction. To define it, beyond explicit substitution (shortened to ES) constructor we need the following contexts with ESs.

$$\begin{array}{rrcl} \lambda\text{-TERMS WITH ESS} & t,u,r & ::= & x \mid \lambda x.t \mid tu \mid t[x\leftarrow u] \\ \text{SUBSTITUTION CONTEXTS} & S,S' & ::= & \langle\cdot\rangle \mid S[x\leftarrow u] \\ \text{(MICRO) WEAK HEAD CONTEXTS} & W,W' & ::= & \langle\cdot\rangle \mid Wu \mid W[x\leftarrow u] \end{array}$$

An explicit substitution $t[x\leftarrow u]$ is an annotation for a meta-level substitution $t\{x\leftarrow u\}$ which has been delayed. As such, it binds x in t but not in u. As it is standard for pen-and-paper reasoning, we work silently modulo α-equivalence, thus in a term such as $(xy)[x\leftarrow xy]$ the left and right occurrences of x are not occurrences of the same variable, as the left one is bound and thus its name is not really x, given that $(xy)[x\leftarrow xy] =_\alpha (zy)[z\leftarrow xy]$.

Substitution contexts are simply lists of ESs. Weak head contexts are the extension to ESs of the applicative contexts given above.

Micro weak head reduction $\to_{mwh}$ is the union of two rules, *β at a distance* $\to_{dB}$ and *micro substitution* $\to_{ms}$, which are usually defined *contextually*. A peculiar aspect of their definition is that contexts are used also in the root case.

$$\text{(Weak head) Beta at a distance} \qquad \frac{}{S\langle\lambda x.t\rangle u \mapsto_{dB} S\langle t[x{\leftarrow}u]\rangle} \qquad \frac{t \mapsto_{dB} u}{W\langle t\rangle \to_{dB} W\langle u\rangle}$$

$$\text{(Weak head) Micro substitution} \qquad \frac{}{W\langle\!\langle x\rangle\!\rangle[x{\leftarrow}u] \mapsto_{ms} W\langle\!\langle u\rangle\!\rangle[x{\leftarrow}u]} \qquad \frac{t \mapsto_{ms} u}{W\langle t\rangle \to_{ms} W\langle u\rangle}$$

Examples for β at a distance: $(\lambda x.t)[y{\leftarrow}u]r \mapsto_{dB} t[x{\leftarrow}r][y{\leftarrow}u]$ and $(\lambda x.t)[y{\leftarrow}u]r[z{\leftarrow}w] \to_{dB} t[x{\leftarrow}r][y{\leftarrow}u][z{\leftarrow}w]$. The word *distance* refers to the presence of ESs – in fact the substitution context S – between the abstraction and the argument, which then interact *at a distance*. Examples for micro substitution: $((xx)t)[y{\leftarrow}u][x{\leftarrow}r] \mapsto_{ms} ((rx)t)[y{\leftarrow}u][x{\leftarrow}r]$ and $((xx)t)[y{\leftarrow}u][x{\leftarrow}r]w \to_{ms} ((rx)t)[y{\leftarrow}u][x{\leftarrow}r]w$. The word *micro* refers to the fact that the rule replaces one occurrence of x at a time. Note the other occurrence of x in the example, which is left untouched. Note also that the root rule $\mapsto_{ms}$ uses the capture-avoiding form of plugging, while the contextual closure defining $\to_{ms}$ uses the capture-allowing variant.

We give alternative inductive definitions of $\mapsto_{dB}$ / $\mapsto_{ms}$, and of their contextual closures, that shall be used in the discussion on how to formalize $\to_{mwh}$ in Abella, in the next section.

$$\text{Inductive def of } \mapsto_{dB} \qquad \frac{}{(\lambda x.t)u \mapsto_{dB} t[x{\leftarrow}u]} \qquad \frac{tu \mapsto_{dB} r}{t[x{\leftarrow}w]u \mapsto_{dB} r[x{\leftarrow}w]}$$

$$\text{Inductive def of } \mapsto_{ms} \qquad \frac{}{x[x{\leftarrow}u] \mapsto_{ms} u[x{\leftarrow}u]} \qquad \frac{t[x{\leftarrow}u] \mapsto_{ms} u[x{\leftarrow}u]}{(tr)[x{\leftarrow}u] \mapsto_{ms} (ur)[x{\leftarrow}u]}$$

$$\frac{t[x{\leftarrow}u] \mapsto_{ms} u[x{\leftarrow}u]}{t[y{\leftarrow}r][x{\leftarrow}u] \mapsto_{ms} u[y{\leftarrow}r][x{\leftarrow}u]}$$

$$\text{Ctx closure of } a \in \mapsto_{dB}, \mapsto_{ms} \qquad \frac{t \mapsto_a u}{t \to_a u} \qquad \frac{t \to_a u}{tr \to_a ur} \qquad \frac{t \to_a u}{t[y{\leftarrow}r] \to_a u[y{\leftarrow}r]}$$

Reachable Terms. Since, in the context of this paper, micro weak head reduction is just a way to refine weak head reduction, we shall consider only terms with ESs that are reachable from a term without ESs by $\to_{mwh}$. It is easily seen that the following characterization holds.

▶ **Definition 2** (Reachable terms). *Reachable terms (possibly with ESs) are defined as follows*

$$\frac{}{x \text{ is reachable}} \qquad \frac{t \text{ is reachable} \qquad u \text{ has no ESs}}{tu \text{ is reachable}}$$

$$\frac{\lambda x.t \text{ has no ESs}}{\lambda x.t \text{ is reachable}} \qquad \frac{t \text{ is reachable} \qquad u \text{ has no ESs}}{t[x{\leftarrow}u] \text{ is reachable}}$$

▶ **Lemma 3.** *If t has no ESs and $t \to^*_{mwh} u$ then u is reachable.*

Processes. The dialect of the π-calculus that we adopt contains only the constructs needed to represent the λ-calculus. Namely, we use the *asynchronous* π-calculus (thus outputs are not prefixes) with both unary and binary inputs and outputs, and pairing up unary inputs with replication. The grammar is:

$$\text{Processes} \quad P, Q, R \quad ::= \quad 0 \mid \overline{x}\langle y\rangle \mid \overline{x}\langle y, z\rangle \mid \nu x P \mid x(y,z).P \mid !x(y).P \mid P \mid Q$$

For channels, we use the same notation that we use for the variables of λ-terms. We postpone the definition of structural equivalence and of the rewriting rules for processes to Sect. 4.

3 Our Approach to Formalizing λ-Terms, Processes, and Contexts

Reasoning Level. Abella has two layers, the *specification level* and the *reasoning level*. They are based on different logics, the reasoning level being more powerful, having in particular the ∇ (nabla) quantifier, and provided with special tactics to reason about the specification level. In many formalizations in Abella, definitions are given at the specification level while statements and proofs are given at the reasoning level. We follow another approach, giving the definitions at the reasoning level. One of the reasons is that in this way we can exploit ∇ to formalize terms with free variables, obtaining definitions, statements, and reasoning that are closer to those with pen and paper. The same approach is used also (at least) by Tiu and Miller [43], Accattoli [1], Chaudhuri et al. [17], and section 7.3 of the Abella tutorial by Baelde et al. [11].

In this paper, we show Abella code to explain how crucial concepts are formalized, but, for lack of space, we do not systematically pair every definition/statement with its code (for the link to the code see the first page, after the abstract). We also assume basic familiarity with the representation of binders in higher-order abstract formalisms (the one adopted by Abella is Miller and Nadathur's λ-*tree syntax* [30]).

Induction on Types in Abella. In Abella, it is standard to define untyped λ-terms by introducing a type `tm` and two constructors for applications and abstractions, without specifying variables, as follows:

```
Kind    tm    type.
Type    app   tm -> tm -> tm.
Type    abs   (tm -> tm) -> tm.
```

Note that `abs` takes an argument of type `tm -> tm` which is how Abella encodes binders. More precisely, an object-level binding constructor such as $\lambda x.t$ is encoded via a pair: an ordinary constructor `abs` applied to a *meta-level abstraction* of type `tm -> tm`. For example, the term $\lambda x.xx$ that binds x in the scope xx is encoded as `abs x\app x x` (that is parsed as `abs (x\app x x)`) where `x\app x x` is a meta-level abstraction of type `tm -> tm` in the syntax of Abella. In the rest of the paper, with an abuse of terminology, we call *binders* such terms of type `tm -> tm`.

Reasoning by induction on the structure of `tm` terms is not possible in Abella because of the *open world assumption*, stating that new constructors can always be added later to any type. Thus, one rather defines a `is_tm` predicate, as follows, and reasons by induction over it:

```
Define is_tm : tm -> prop by
    nabla x, is_tm x;
    is_tm (abs T) := nabla x, is_tm (T x);
    is_tm (app T U) := is_tm T /\ is_tm U.
```

The first clause uses nabla to say that the free variable `x` is a term. Variables with capitalized names in the last two clauses are implicitly quantified by $\forall$ at the clause level. The second

clause states that an abstraction `abs T` is a term if its body is a term. The body is obtained applying the binder `T` (of type `tm -> tm`) to a fresh variable `x` to obtain a term of type `T`. Such application corresponds in a pen-and-paper proof to the (usually implicit) logical step that replaces the bound variable with a fresh one.

Predicates such as `is_tm` might seem an oddity. In our development, we exploit them crucially, as we now explain.

One Type to Formalize Them All. We deal with three main syntactic categories, ordinary λ-terms, λ-terms with ESs, and processes. In order to circumvent the issue of relating term variables and process names, we formalize the three categories over the same type `pt`, standing for *processes and terms*, and then distinguish them via dedicated predicates. The constructors are defined as follows:

```
Kind pt type.

% terms
Type abs (pt -> pt) -> pt.
Type app pt -> pt -> pt.
Type esub (pt -> pt) -> pt -> pt.
```

```
% pi-calculus terms
Type zero pt.
Type nu (pt -> pt) -> pt.
Type par pt -> pt -> pt.
Type out pt -> pt -> pt.
Type out2 pt -> pt -> pt -> pt.
Type in pt -> (pt -> pt) -> pt.
Type in2 pt -> (pt -> pt -> pt) -> pt.
```

About terms, `esub T U` represents $t[x \leftarrow u]$. For processes, `nu` is the restriction operator $\nu x P$, `par` is parallel composition $P \mid Q$, `out` and `out2` are unary and binary output, and similarly for inputs. Note the binders used by the input prefixes: `in2 x P` represents $x(y,z).P$, where y and z are implicit in the fact that `P` has type `pt -> pt -> pt`. They can be made explicit using the equivalent representation `in2 x (y\z\ P y z)`, where `y\z\P y z` is an η-expansion of `P`: Abella identifies all meta-level abstractions up to α-renaming and η-expansion.

We then use two predicates, one for isolating λ-terms with no ESs and one for reachable λ-terms with ESs – processes shall not need one, as there shall be no inductions on processes.

```
Define tm_with_no_ES : pt -> prop by
  nabla x, tm_with_no_ES x;
  tm_with_no_ES (abs T) := nabla x, tm_with_no_ES (T x);
  tm_with_no_ES (app T U) := tm_with_no_ES T /\ tm_with_no_ES U.

Define reachable_tm : pt -> prop by
  nabla x, reachable_tm x;
  reachable_tm (abs T) := tm_with_no_ES (abs T);
  reachable_tm (app T U) := reachable_tm T /\ tm_with_no_ES U;
  reachable_tm (esub T U) := (nabla x, reachable_tm (T x)) /\ tm_with_no_ES U.
```

Micro Weak Head Reduction. For the formal definitions of the two rules of $\to_{mwh}$ we explored various approaches. We started with the monolithic approach, but soon realized that it was very difficult, if not impossible, to reason about the properties of the rules. We then adopted a split approach. For both $\mapsto_{dB}$ and $\to_{dB}$ (`red_root_db` and `red_db` in the sources), we use the inductive definitions, because their use of context plugging is capture-allowing. For both $\mapsto_{ms}$ and $\to_{ms}$, we initially used the inductive definitions, which is enough to prove the relationship with the π-calculus. To prove the relationship between $\to_{wh}$ and $\to_{mwh}$, which requires a fine analysis of $\mapsto_{ms}$ / $\to_{ms}$ steps, we were however led

to switch (in the whole development) to a mixed style: contextual definition of $\mapsto_{ms}$, which enables finer reasoning, and inductive definition of $\to_{ms}$. This is possible because – crucially – the definition of $\mapsto_{ms}$ uses capture-avoiding plugging $W\langle\!\langle u \rangle\!\rangle$. The idea is that weak head contexts W can be formalized as follows (that is, as a function $u \mapsto W\langle\!\langle u \rangle\!\rangle$).

```
Define weak_head_ctx : (pt -> pt) -> prop by
    weak_head_ctx (h\h);
    weak_head_ctx (h\ app (W h) U) := weak_head_ctx  W;
    weak_head_ctx (h\ esub (x\ W x h) U) := nabla x, weak_head_ctx (W x).
```

We use `h` for *hole*. In the last clause, `h\ esub (x\ W x h) U` specifies that the explicit substitution bounds a variable x which is not the one representing the hole.

We now describe the definition of $\mapsto_{ms}$, based on two predicates, avoiding Abella code for lack of space, but reflecting the Abella formalization faithfully.

Firstly, we need the predicate t *has free head variable* x, shortened $\mathsf{fhv}(t) = x$, defined by:

$$\frac{}{\mathsf{fhv}(x) = x} \qquad \frac{\mathsf{fhv}(t) = x}{\mathsf{fhv}(tu) = x} \qquad \frac{\mathsf{fhv}(t) = x}{\mathsf{fhv}(t[y{\leftarrow}u]) = x}$$

Secondly, the predicate t *has free maximal weak head context* W, shortened $\mathsf{maxw}(t) = W$, defined by:

$$\frac{}{\mathsf{maxw}(x) = \langle\cdot\rangle} \qquad \frac{}{\mathsf{maxw}(\lambda x.t) = \langle\cdot\rangle} \qquad \frac{\mathsf{maxw}(t) = W}{\mathsf{maxw}(tu) = Wu} \qquad \frac{\mathsf{maxw}(t) = W}{\mathsf{maxw}(t[x{\leftarrow}u]) = W[x{\leftarrow}u]}$$

An immediate lemma guarantees that if $\mathsf{fhv}(t) = x$ then there exists a weak head context W such that $\mathsf{maxw}(t) = W$. Now, the micro substitution root rule $\mapsto_{ms}$ (`red_root_ms` in the sources) is defined as follows:

$$\frac{\mathsf{fhv}(t) = x \text{ and } \mathsf{maxw}(t) = W}{t[x{\leftarrow}u] \;\mapsto_{ms}\; W\langle\!\langle u \rangle\!\rangle[x{\leftarrow}u]}$$

Finally, the inductive contextual closure that defines $\to_{dB}$ (`red_db`) and $\to_{ms}$ (`red_ms`) is obtained via a higher-order predicate `ctx_cl_tm` taking a relation and returning its contextual closure defined as follows, where $\mathsf{a} \in \{dB, ms\}$.

$$\frac{t \mapsto_{\mathsf{a}} u}{t \to_{\mathsf{a}} u} \qquad \frac{t \to_{\mathsf{a}} u}{tr \to_{\mathsf{a}} ur} \qquad \frac{t \to_{\mathsf{a}} u}{t[x{\leftarrow}r] \to_{\mathsf{a}} u[x{\leftarrow}r]}$$

Note that it is not possible to use a contextual formalization of the contextual closure, because the closure rests on capture-allowing plugging, which is not supported by Abella.

Finally, micro weak head reduction $\to_{mwh}$ (`red_mwh`) is defined as $\to_{dB} \cup \to_{ms}$.

4 The π-Calculus, at a Distance

Here we describe the formalization of the rewriting rules of the π-calculus. We start with structural equivalence and give the standard presentation of the rules. Then, we redefine them according to the *at a distance* approach developed by Accattoli [2].

Structural Equivalence. Processes are considered modulo structural congruence $\equiv$, defined in two steps. First, we define *root structural equivalence* $P \doteq Q$ (`str_eq_root`) via the following clauses.

- *Neutrality of* 0 *with respect to parallel composition*: $P \mid 0 \doteq P$, $0 \mid P \doteq P$, $P \doteq P \mid 0$, and $P \doteq 0 \mid P$;

- *Irrelevance of vacuous name restrictions*: $\nu x P \doteq P$, and $P \doteq \nu x P$ if $x \notin \mathsf{fn}(P)$[1];
- *Commutativity of parallel composition*: $P \mid Q \doteq Q \mid P$
- *Associativity of parallel composition*: $P \mid (Q \mid R) \doteq (P \mid Q) \mid R$ and $(P \mid Q) \mid R \doteq P \mid (Q \mid R)$;
- *Permutation of name restriction and parallel composition*: $\nu x(P \mid Q) \doteq P \mid \nu x Q$ and $P \mid \nu x Q \doteq \nu x(P \mid Q)$ if $x \notin \mathsf{fv}(P)$, $\nu x(P \mid Q) \doteq \nu x P \mid Q$ and $\nu x P \mid Q \doteq \nu x(P \mid Q)$ if $x \notin \mathsf{fv}(Q)$;
- *Commutativity of name restrictions*: $\nu x \nu y P \doteq \nu y \nu x P$.

Note that all clauses but the commutativity ones have symmetric clauses. It is natural to wonder whether is it possible to have only half of the clauses plus a rule for the symmetry of $\doteq$. Such an alternative does not seem to work, as it shall be explained in the next section.

The second step is defining structural equivalence $\equiv$ (`str_eq`) as the equivalence and contextual closure of $\doteq$, what is sometimes called *the congruence closure*. The symmetric definition of $\doteq$ spares us a symmetry rule in the definition of $\equiv$, as proved after the definition.

Defining structural equivalence on top of root structural equivalence

$$\frac{P \doteq Q}{P \equiv Q} \qquad \frac{}{P \equiv P} \qquad \frac{P \equiv Q \qquad Q \equiv R}{P \equiv R}$$

$$\frac{P \equiv Q}{\nu x P \equiv \nu x Q} \qquad \frac{P \equiv Q}{P \mid R \equiv Q \mid R} \qquad \frac{P \equiv Q}{R \mid P \equiv R \mid Q}$$

▶ **Lemma 4.**

1. Symmetry of $\doteq$: *if* $P \doteq Q$ *then* $Q \doteq P$.
2. Symmetry of $\equiv$: *if* $P \equiv Q$ *then* $Q \equiv P$.

Proof. Point 1 is by case analysis: every clause of $\doteq$ is either symmetric or has a symmetric clause. Point 2 is by induction on $P \equiv Q$, using Point 1 in the case for $\doteq$. ◀

The Ordinary Rewriting Rules. The π-calculus at work here has two rewriting rules, a linear one for binary communication and a rule involving process replication for unary communication. The root binary and replication rules are defined as follows (`ord_pi-red_root_bin P Q` and `ord_pi-red_root_rep P Q` in Abella).

$$\overline{x}\langle y, z\rangle \mid x(y', z').Q \mapsto_{bin} Q\{y' \leftarrow y\}\{z' \leftarrow z\} \qquad \overline{x}\langle y\rangle \mid !x(z).Q \mapsto_{!} Q\{z \leftarrow y\} \mid !x(z).Q$$

The root rules are closed contextually under restrictions and parallel composition as follows.

Contextual closure of root rules on processes, $a \in \{bin, !\}$

$$\frac{P \mapsto_a Q}{\nu x P \to_a \nu x Q} \qquad \frac{P \to_a Q}{\nu x P \to_a \nu x Q} \qquad \frac{P \to_a Q}{P \mid R \to_a Q \mid R} \qquad \frac{P \to_a Q}{R \mid P \to_a R \mid Q} \tag{1}$$

In Abella, the closure is realised via a higher-order closure predicate `ctx_cl_pr`, similarly to what is done for micro weak head reduction.

The rules at work in the π-calculus are actually $\to_{bin}$ and $\to_{!}$ *modulo* $\equiv$, that is, one rather considers the rules $\to_{bin/\equiv}$ and $\to_{!/\equiv}$ where $P \to_{bin/\equiv} Q$ if there exist P' and Q' such that $P \equiv P' \to_{bin} Q' \equiv Q$ (one might also compactly write $\to_{bin/\equiv} := \equiv \to_{bin} \equiv$) and similarly for $\to_{!/\equiv}$.

[1] In [2], Accattoli uses $\nu x 0 \doteq 0$, and $0 \doteq \nu x 0$, which is correct, but then requires one to prove the general version as an easy lemma.

Towards Communication at a Distance. The use of structural equivalence in the definition of the rewriting relation of the π-calculus induces some annoying complications when one tries to reflect process reductions on terms. We are then going to reformulate the π-calculus reduction rules *at a distance*, that is, in a way that allows us to prove a postponement theorem with respect to structural equivalence, inspired by Accattoli [2] but in a slightly different way. Let us recall the idea from [2].

The first step is to define *non-blocking contexts*, which are the contexts used in the contextual closure in (1), as follows:

$$\text{Non-blocking ctxs} \quad N, M \quad ::= \quad \langle\cdot\rangle \;\mid\; N \,|\, Q \;\mid\; P \,|\, N \;\mid\; \nu x N$$

The second step is to generalize the root case of, say, the binary rule, as follows:

$$\frac{N \text{ and } M \text{ do not capture } x}{N\langle \overline{x}\langle y, z\rangle\rangle \,|\, M\langle x(y', z').Q\rangle \;\mapsto_{bin}\; N\langle M\langle Q\{y'{\leftarrow}y\}\{z'{\leftarrow}z\}\rangle\rangle} \tag{2}$$

and then one closes the root rule by contexts as in (1). To be precise, since the aim is to avoid structural equivalence in rewriting steps, one also needs the symmetric case of (2). The idea behind the postponement is that the role of structural equivalence $\equiv$ in the ordinary approach is to re-organize the term as to move the non-blocking contexts N and M out of the way, that is, as to obtain:

$$N\langle \overline{x}\langle y, z\rangle\rangle \,|\, M\langle x(y', z').Q\rangle \equiv N\langle M\langle \overline{x}\langle y, z\rangle \,|\, x(y', z').Q\rangle\rangle$$

to then allow one to apply the ordinary communication rule. The approach at a distance avoids the re-organization altogether, by defining communication *up to* non-blocking contexts.

Here, we slightly refine the presented idea, in two respects. Firstly, the plugging at work in (2) is capture-allowing (in contrast to the root case of $\mapsto_{ms}$ for terms), so that we replace it with an inductive contextual closure. Secondly, the scheme in (2) is *asymmetric*: in the reduct, the contexts are composed as $N\langle M\rangle$, while the opposite composition would work as well. By turning to an inductive contextual closure, we can restore the symmetry. In fact, we obtain a strictly more permissive rule, as we permit the reduct to have any shuffling of the constructors in N and M. The rule is then non-deterministic, but harmlessly so, as all the reductions of a same redex are structurally equivalent.

Communication at a Distance. The rule at a distance for binary prefixes $\Rrightarrow_{bin}$ is obtained by first defining its *root* variant $\Rightarrow_{bin}$ (`new_pi-red_root_bin P Q` in Abella), as follows.

$$\frac{}{\overline{x}\langle y, z\rangle \,|\, x(y', z').Q \;\Rightarrow_{bin}\; Q\{y'{\leftarrow}y\}\{z'{\leftarrow}z\}} \qquad \frac{}{x(y', z').Q \,|\, \overline{x}\langle y, z\rangle \;\Rightarrow_{bin}\; Q\{y'{\leftarrow}y\}\{z'{\leftarrow}z\}}$$

$$\frac{P \,|\, Q \Rightarrow_{bin} R}{(P \,|\, O) \,|\, Q \Rightarrow_{bin} R \,|\, O} \qquad \frac{P \,|\, Q \Rightarrow_{bin} R}{P \,|\, (Q \,|\, O) \Rightarrow_{bin} R \,|\, O} \qquad \frac{P \,|\, Q \Rightarrow_{bin} R}{\nu x P \,|\, Q \Rightarrow_{bin} \nu x R}$$

$$\frac{P \,|\, Q \Rightarrow_{bin} R}{(O \,|\, P) \,|\, Q \Rightarrow_{bin} O \,|\, R} \qquad \frac{P \,|\, Q \Rightarrow_{bin} R}{P \,|\, (O \,|\, Q) \Rightarrow_{bin} O \,|\, R} \qquad \frac{P \,|\, Q \Rightarrow_{bin} R}{P \,|\, \nu x Q \Rightarrow_{bin} \nu x R}$$

Then, $\Rrightarrow_{bin}$ (`new_pired_bin P Q` in Abella) is obtained by applying the same context closure as in (1) to $\Rightarrow_{bin}$.

The rule for unary prefixes $\Rrightarrow_!$ is obtained via the same construction by simply changing the base case. The root case $\Rightarrow_!$ (`new_pi-red_root_rep P Q` in Abella) follows.

$$\frac{}{\overline{x}\langle y\rangle \,|\, !x(z).Q \mapsto_! Q\{z{\leftarrow}y\} \,|\, !x(z).Q} \qquad \frac{}{!x(z).Q \,|\, \overline{x}\langle y\rangle \mapsto_! !x(z).Q \,|\, Q\{z{\leftarrow}y\}}$$

$$\frac{P \,|\, Q \Rrightarrow_! R}{(P \,|\, O) \,|\, Q \Rrightarrow_! R \,|\, O} \qquad \frac{P \,|\, Q \Rrightarrow_! R}{P \,|\, (Q \,|\, O) \Rrightarrow_! R \,|\, O} \qquad \frac{P \,|\, Q \Rrightarrow_! R}{\nu x P \,|\, Q \Rrightarrow_! \nu x R}$$

$$\frac{P \,|\, Q \Rrightarrow_! R}{(O \,|\, P) \,|\, Q \Rrightarrow_! O \,|\, R} \qquad \frac{P \,|\, Q \Rrightarrow_! R}{P \,|\, (O \,|\, Q) \Rrightarrow_! O \,|\, R} \qquad \frac{P \,|\, Q \Rrightarrow_! R}{P \,|\, \nu x Q \Rrightarrow_! \nu x R}$$

The general rule (`new_pi-red_rep P Q` in Abella) is then obtained by a contextual closure, as for $\Rrightarrow_{bin}$. Finally, we set $\Rrightarrow := \Rrightarrow_{bin} \cup \Rrightarrow_!$. Some basic properties of reduction follow.

▶ **Lemma 5.** *Let $a \in \{bin, !\}$.*
1. No creation of free names*: if $P \Rrightarrow_a Q$ then $\mathsf{fv}(Q) \subseteq \mathsf{fv}(P)$.*
2. Parallel symmetry*: if $P \,|\, Q \Rightarrow_a R$ then exists O such that $Q \,|\, P \Rightarrow_a O$ and $O \equiv R$.*

5 Postponement of $\equiv$ and Equivalence of the Presentations

The main property of the presentation at a distance is that $\equiv$ strongly postpones with respect to $\Rrightarrow_{bin}$ and $\Rrightarrow_!$, that is, it is not required for reduction. A noticeable point of the proof is that the two cases of $\Rrightarrow_{bin}$ and $\Rrightarrow_!$ are handled in the *exact* same way: all the statements and all the proofs are indeed parametric in the reduction rule (the Abella proofs are identically structured but not proved parametrically). We need two auxiliary lemmas.

▶ **Lemma 6** (Auxiliary properties for postponement of structural equivalence)**.** *Let $a \in \{bin, !\}$.*
1. *If $(P \,|\, Q) \,|\, R \Rightarrow_a O$ then (exists O' such that $P \,|\, R \Rightarrow_a O'$ and $O \equiv O' \,|\, Q$) or (exists O' such that $Q \,|\, R \Rightarrow_a O'$ and $O \equiv P \,|\, O'$).*
2. *If $\nu x P \,|\, Q \Rightarrow_a R$ then there exists O such that $P \,|\, Q \Rightarrow_a O$ and $\nu x O \equiv R$.*

Proof. Every point is by induction on the $\Rightarrow_a$ step in its hypothesis. In Point 1, for $a = bin$ one actually has $O = R$. ◀

The lemma allows us to prove the postponement in the root case of structural equivalence, that we prefer to isolate to stress that it does not need an induction and because it does not need the lemma that follows it.

▶ **Proposition 7** (Root strong postponement of $\equiv$ wrt $\Rrightarrow$)**.** *Let $a \in \{bin, !\}$. If $P \doteq P'$ and $P \Rrightarrow_a Q$ then there exists Q' such that $P' \Rrightarrow_a Q'$ and $Q' \equiv Q$.*

Proof. By case analysis of $P \doteq P'$, using Lemma 5 and Lemma 6. ◀

To deal with the general case of structural equivalence, we need a further auxiliary lemma.

▶ **Lemma 8.** *Let $a \in \{bin, !\}$. If $P \equiv P'$ and $P \,|\, Q \Rrightarrow_a R$ then there exists R' such that $P' \,|\, Q \Rrightarrow_a R'$ and $R \equiv R'$.*

Proof. By case analysis of $P \doteq P'$, using Lemma 5 and Lemma 6. ◀

▶ **Theorem 9** (Strong postponement of $\equiv$ wrt $\Rrightarrow$)**.** *Let $a \in \{bin, !\}$. If $P \equiv P'$ and $P \Rrightarrow_a Q$ then there exists Q' such that $P' \Rrightarrow_a Q'$ and $Q' \equiv Q$.*

Proof. By induction on $P \equiv P'$. The case for $\doteq$ is exactly Prop. 7. The case of reflexivity is immediate. The case of contextual closure with respect to parallel composition on the right uses Lemma 8, and the case on the left uses Lemma 8 and Lemma 5.2. The other cases (transitivity and name restriction closure) follow by the *i.h.* ◀

▶ Remark 10. We can now explain why adding a symmetry rule to the definition of $\equiv$ (or $\doteq$), and so dividing by 2 the number of cases defining $\doteq$, does not work. Consider the proof of the strong postponement property for the symmetry case: the hypotheses of the theorem are $P \equiv P'$ and $P \Rrightarrow_a Q$, and the inductive case we are facing is that $P \equiv P'$ because $P' \equiv P$. To be able to apply the *i.h.* one should have $P' \Rrightarrow_a R$ for some R, which is what we actually have to prove. Note also that $P' \Rrightarrow_a R$ is not enough because the *i.h.* would then give a process O which is not necessarily Q. Therefore, to prove the statement one should have $P' \Rrightarrow_a Q$, which is not even true.

Equivalence of Presentations. The strong postponement property is the key point in showing that $\Rrightarrow_a \equiv$ is equivalent to $\rightarrow_{a/\equiv}$ for $a \in \{bin, !\}$.

▶ **Lemma 11.** *Let $a \in \{bin, !\}$. If $P \rightarrow_a Q$ then $P \Rrightarrow_a Q$.*

▶ **Theorem 12** (Ordinary and new presentations are equivalent)**.** *Let $a \in \{bin, !\}$.*
1. *If $P \rightarrow_{a/\equiv} Q$ then there exists R such that $P \Rrightarrow_a R \equiv Q$.*
2. *If $P \Rightarrow_a Q$ then $P \rightarrow_{a/\equiv} Q$.*
3. *If $P \Rrightarrow_a Q$ then $P \rightarrow_{a/\equiv} Q$.*

Proof.
1. Explicitly, $P \equiv P' \rightarrow_a Q' \equiv Q$. By Lemma 11, $P' \Rrightarrow_a Q'$. By strong postponement (Theorem 9), there exists R such that $P \Rrightarrow_a R \equiv Q'$, and, by transitivity of structural equivalence, $P \Rrightarrow_a R \equiv Q$.
2. Immediate induction on $P \Rightarrow_a Q$.
3. Immediate induction on $P \Rrightarrow_a Q$, using the previous point in the base case. ◀

6 Translation and Simulations

Milner's Translation. Here we present Milner's call-by-name translation $[\![t]\!]$ of the λ-calculus to the π-calculus, extended to account also for ESs. Let $a, b, c, \ldots$ be *special channel names*. Milner presented the translation as $[\![t]\!]_a$, that is, as parametrized by a special channel name a, meant to be the channel on which t itself can be communicated. Sometimes, typically by Sangiorgi, the translation is rather presented moving the parametrization on the π-calculus side, stating that the representation of a λ-term (with ESs) is a function $\lambda\!\!\lambda a.P$ (where $\lambda\!\!\lambda a.$ is a meta-level notation not part of the syntax of processes) which when applied to b gives the process $(\lambda\!\!\lambda a.P)_b = P\{a \leftarrow b\}$. While both approaches are viable (and we explored both), we prefer Sangiorgi's. Firstly, it can easily be represented in Abella, by seeing the functions $\lambda\!\!\lambda a.P$ as process binders. Secondly, it reduces the amount of free names that have to be managed (too many free names make Abella produce unreadable intermediate goals during the formalization). The translation is then defined as follows.

Translation of terms with ESs to processes

$$
\begin{array}{rcll}
[\![x]\!] & := & \lambda\!\!\lambda a.\overline{x}\langle a\rangle & \\
[\![\lambda x.t]\!] & := & \lambda\!\!\lambda a.a(x,b).[\![t]\!]_b & a \notin \mathsf{fv}([\![t]\!]_b) \\
[\![tu]\!] & := & \lambda\!\!\lambda a.\nu b \nu x([\![t]\!]_b \mid \overline{b}\langle x, a\rangle \mid\, !x(c).[\![u]\!]_c) & a, b \notin \mathsf{fv}(!x(c).[\![u]\!]_c) \\
 & & & a \notin \mathsf{fv}([\![t]\!]_b), x \notin \mathsf{fv}(t) \cup \mathsf{fv}(u) \\
[\![t[x \leftarrow u]]\!] & := & \lambda\!\!\lambda a.\nu x([\![t]\!]_a \mid\, !x(b).[\![u]\!]_b) & a \notin \mathsf{fv}([\![u]\!]_b)
\end{array}
$$

Beyond the given side conditions, the translation rests on the assumption that the special names do not occur as variables of terms. Note that the subjects of binary input/outputs are always special names, while the subjects of unary input/outputs are variable names. We

distinguish special names from variables for readability and because the distinction helps us understand the translation. There is no formal need to distinguish them, however, and indeed our Abella formalization, which follows, does not use a separate category for them.

```
Define translation : pt -> (pt -> pt) -> prop by
  nabla x, translation x (out x);
  translation (abs T) (a\in2 a P) := nabla x, translation (T x) (P x);
  translation (app T U) (a\nu b\nu x\par (P b) (par (out2 b x a) (in x Q)))
      := translation T P /\ translation U Q;
  translation (esub T U) (a\nu x\par (P x a) (in x Q))
      := (nabla x, translation (T x) (P x)) /\ translation U Q.
```

Note the use of `out x` in the translation of a variable x. The type of `out x` is `pt -> pt -> pt`, so the unary output $\overline{x}\langle a\rangle$ is represented by `out x a`. But the translation uses `out x` instead: this is done to map x to the function (or binding) $\lambda\!\!\lambda a.\overline{x}\langle a\rangle$ rather than to $\overline{x}\langle a\rangle$. Similar remarks apply to the other cases. Note also that the Abella definition states the side conditions *dually*, by saying where variables *can* appear, rather than where they do not, *e.g.* `b` can appear in `P b` but not in `Q`, in the application case.

π Simulates $\to_{mwh}$. A single step of the micro weak head reduction rules $\to_{dB}$ and $\to_{ms}$ is simulated in the π-calculus by exactly one step of $\Rrightarrow_{bin}$ and $\Rrightarrow_!$, respectively, followed by structural equivalence. We detail the case of $\to_{ms}$, mimicking closely the proof in Abella. One needs an auxiliary lemma essentially capturing the simulation of the root case $\mapsto_{ms}$ by the root case $\Rightarrow_!$, and then the simulation extends smoothly to the contextual closures.

▶ **Lemma 13** (Auxiliary lemma for $\mapsto_{ms}$). *Let* $\mathsf{fhv}(t) = x$, $\mathsf{maxw}(t) = W$, $[\![t[x{\leftarrow}u]]\!] = \lambda\!\!\lambda a.\nu x P$, *and* $[\![W\langle u\rangle[x{\leftarrow}u]]\!] = \lambda\!\!\lambda a.\nu x Q$. *Then there exists* R *such that* $P \Rightarrow_! R$ *and* $R \equiv Q$.

Proof. By induction on $\mathsf{fhv}(t) = x$. Note that $[\![t[x{\leftarrow}u]]\!] = \lambda\!\!\lambda a.\nu x([\![t]\!]_a \mid !x(b).[\![u]\!]_b)$ and $[\![W\langle u\rangle[x{\leftarrow}u]]\!] = \lambda\!\!\lambda a.\nu x([\![W\langle u\rangle]\!]_a \mid !x(b).[\![u]\!]_b)$, thus we have to prove that there is R such that

$$P = [\![t]\!]_a \mid !x(b).[\![u]\!]_b \quad \Rightarrow_! \quad R \quad \equiv \quad [\![W\langle u\rangle]\!]_a \mid !x(b).[\![u]\!]_b = Q.$$

Cases of $\mathsf{fhv}(t) = x$:

1. *Head variable*, that is, $t = x$. Then $W = \langle\cdot\rangle$ and $W\langle u\rangle = u$. We have:
$$P \quad = \quad [\![x]\!]_a \mid !x(b).[\![u]\!]_b \quad = \quad \overline{x}\langle a\rangle \mid !x(b).[\![u]\!]_b \quad \Rightarrow_! \quad [\![u]\!]_a \mid !x(b).[\![u]\!]_b \quad = \quad Q$$
2. *Left of application*, that is, $t = rw$ with $\mathsf{fhv}(r) = x$. Then $W = W'w$ and $W\langle u\rangle = W'\langle u\rangle w$. By *i.h.*, there exists R' such that
$$[\![r]\!]_a \mid !x(b).[\![u]\!]_b \quad \Rightarrow_! \quad R' \quad \equiv \quad [\![W'\langle u\rangle]\!]_a \mid !x(b).[\![u]\!]_b.$$
Then:
$$\begin{array}{rcll} P & = & [\![rw]\!]_a \mid !x(b).[\![u]\!]_b & = \\ & & \nu c\nu y([\![r]\!]_c \mid \overline{c}\langle y, a\rangle \mid !y(d).[\![w]\!]_d) \mid !x(b).[\![u]\!]_b & \Rightarrow_! \quad (i.h.) \\ & & \nu c\nu y(R' \mid \overline{c}\langle y, a\rangle \mid !y(d).[\![w]\!]_d) & \equiv \quad (i.h.) \\ & & \nu c\nu y([\![W'\langle u\rangle]\!]_a \mid !x(b).[\![u]\!]_b \mid \overline{c}\langle y, a\rangle \mid !y(d).[\![w]\!]_d) & \equiv \\ & & \nu c\nu y([\![W'\langle u\rangle]\!]_a \mid \overline{c}\langle y, a\rangle \mid !y(d).[\![w]\!]_d) \mid !x(b).[\![u]\!]_b & = \\ & & [\![W\langle u\rangle w]\!]_a \mid !x(b).[\![u]\!]_b & = \quad Q \end{array}$$
3. *Left of substitution*, that is, $t = r[y{\leftarrow}w]$ with $\mathsf{fhv}(r) = x$. Then $W = W'[y{\leftarrow}w]$ and $W\langle u\rangle = W'\langle u\rangle[y{\leftarrow}w]$. By *i.h.*, there exists R' such that
$$[\![r]\!]_a \mid !x(b).[\![u]\!]_b \quad \Rightarrow_! \quad R' \quad \equiv \quad [\![W'\langle u\rangle]\!]_a \mid !x(b).[\![u]\!]_b.$$
Then:

$$
\begin{array}{rcll}
P & = & [\![r[y{\leftarrow}w]]\!]_a \mid !x(b).[\![u]\!]_b & = \\
 & & \nu y([\![t]\!]_a \mid !y(c).[\![w]\!]_c) \mid !x(b).[\![u]\!]_b & \Rrightarrow_! \quad (i.h.) \\
 & & \nu y(R' \mid !y(c).[\![w]\!]_c) & \equiv \quad (i.h.) \\
 & & \nu y([\![W'\langle u\rangle]\!]_a \mid !x(b).[\![u]\!]_b \mid !y(c).[\![w]\!]_c) & \equiv \\
 & & \nu c\nu y([\![W'\langle u\rangle]\!]_a \mid !y(c).[\![w]\!]_c) \mid !x(b).[\![u]\!]_b & = \\
 & & [\![W\langle u\rangle[y{\leftarrow}w]]\!]_a \mid !x(b).[\![u]\!]_b & = \quad Q
\end{array}
$$

◀

▶ **Proposition 14.** *Let t be reachable.*

1. *If $t \mapsto_{ms} u$ then $[\![t]\!] \Rrightarrow_! [\![u]\!]$.*
2. *If $t \to_{ms} u$ then $[\![t]\!] \Rrightarrow_! [\![u]\!]$.*

Proof. For Point 1, note that if $t \mapsto_{ms} u$ then $t = r[x{\leftarrow}w]$ with $\mathsf{fhv}(r) = x$ and $\mathsf{maxw}(r) = W$, so that $u = W\langle w\rangle[x{\leftarrow}w]$. Since the translation of ESs starts with $\lambda\!\!\lambda a.\nu x$, we apply Lemma 13 and obtain $[\![t]\!] \Rightarrow_! [\![u]\!]$. Point 2 is an induction on $t \to_{ms} u$ using Point 1 in the base case. ◀

$\to_{mwh}$ Simulates π. The converse simulation follows exactly the same schema, the root case needs an auxiliary lemma (proof omitted), and then the simulation smoothly lifts to the contextual closures. The only difference, in Abella, is that the case analyses of π-calculus reduction require a few simple lemmas (omitted here) to rule out some impossible cases.

▶ **Lemma 15** (Auxiliary lemma for $\Rrightarrow_!$)**.** *Let $[\![t]\!] = \lambda\!\!\lambda a.P$, $[\![u]\!] = \lambda\!\!\lambda b.Q$. If $P \mid !x(b).Q \Rrightarrow_! R$ then $\mathsf{fhv}(t) = x$ and $\exists$ W and O s.t. $\mathsf{maxw}(t) = W$, $[\![W\langle u\rangle[x{\leftarrow}u]]\!] = \lambda\!\!\lambda a.\nu x O$, and $O \equiv R$.*

▶ **Proposition 16.**

1. *If $[\![t]\!] = \lambda\!\!\lambda a.\nu x P$ and $P \Rrightarrow_! Q$ then $\exists$ u and R s.t. $t \mapsto_{ms} u$, $[\![u]\!] = \lambda\!\!\lambda a.\nu x R$ and $Q \equiv R$.*
2. *If $[\![t]\!] = \lambda\!\!\lambda a.P$ and $[\![t]\!]_a = P \Rrightarrow_! Q$ then $\exists$ u and R s.t. $t \to_{ms} u$, $[\![u]\!] = \lambda\!\!\lambda a.R$ and $Q \equiv R$.*

Proof.

1. For $P \Rrightarrow_! Q$ to hold, t has to be an ES $t = r[x{\leftarrow}w]$ and r and w verify the hypotheses of Lemma 15. Then the conclusions of the lemma are exactly that $r = W\langle x\rangle$, and so $t \mapsto_{\mathsf{ls}} u$ with $u := W\langle w\rangle[x{\leftarrow}w]$, and that there exist processes as in the statement.
2. By induction on $[\![t]\!] = \lambda\!\!\lambda a.P$. Cases:
   - **a.** *Variable*: $t = x$. Impossible because then $P = \overline{x}\langle a\rangle$ which is $\Rrightarrow_!$-normal.
   - **b.** *Abstraction*: $t = \lambda x.r$. Impossible, because $P = a(x,b).[\![t]\!]_b$ is $\Rrightarrow_!$-normal.
   - **c.** *Application*: $t = rw$. Then $P = \nu b\nu x([\![r]\!]_b \mid \overline{b}\langle x,a\rangle \mid !x(c).[\![w]\!]_c)$. Cases of $P \Rrightarrow_! Q$:
     - **i.** *Root step of* $P' = [\![r]\!]_b \mid \overline{b}\langle x,a\rangle \mid !x(c).[\![w]\!]_c \Rrightarrow_! Q'$. Since by definition of the translation $x \notin \mathsf{fv}([\![r]\!]_b)$, there cannot be any root step in P'. In Abella proving this fact requires a couple of straightforward auxiliary lemmas.
     - **ii.** *Inductive*, that is, $P \Rrightarrow_! Q$ because $[\![r]\!]_b \Rrightarrow_! Q'$ for some Q'. By *i.h.*, there exist u' and R' such that $r \to_{\mathsf{ls}} u'$, $[\![u']\!] = \lambda\!\!\lambda b.R'$, and $Q' \equiv R'$. Then $t = rw \to_{\mathsf{ls}} u'w = u$. By applying these equalities to the step $P \Rrightarrow_! Q$ we obtain:
$$
\begin{array}{rcll}
P & = & \nu b\nu x([\![r]\!]_b \mid \overline{b}\langle x,a\rangle \mid !x(c).[\![w]\!]_c) & \\
 & \Rrightarrow_! & \nu b\nu x(Q' \mid \overline{b}\langle x,a\rangle \mid !x(c).[\![w]\!]_c) & \\
(i.h.) & \equiv & \nu b\nu x(R' \mid \overline{b}\langle x,a\rangle \mid !x(c).[\![w]\!]_c) & \\
(i.h.) & \equiv & \nu b\nu x([\![u']\!]_b \mid \overline{b}\langle x,a\rangle \mid !x(c).[\![w]\!]_c) & =: \quad R
\end{array}
$$
     Note that $[\![u]\!] = [\![u'w]\!] = \lambda\!\!\lambda a.R$.
   - **d.** *Substitution*: $t = r[y{\leftarrow}w]$. Then $P = \nu x([\![r]\!]_a \mid !x(b).[\![w]\!]_b)$. Cases of $P \Rrightarrow_! Q$:
     - **i.** *Root step of* $[\![r]\!]_a \mid !x(b).[\![w]\!]_b \Rrightarrow_! Q'$. Then it follows from Point 1.

ii. *Inductive*, that is, $P \Rrightarrow_! Q$ because $[\![r]\!]_a \Rrightarrow_! Q'$ for some Q'. By *i.h.*, there exist u' and R' such that $r \to_{\mathsf{ls}} u'$, $[\![u']\!] = \lambda\!\!\lambda a.R'$, and $Q' \equiv R'$. Then $t = r[y{\leftarrow}w] \to_{\mathsf{ls}} u'[y{\leftarrow}w] = u$. By applying these equalities to the step $P \Rrightarrow_! Q$ we obtain:

$$\begin{array}{rcll} P & = & \nu x([\![r]\!]_a \mid !x(b).[\![w]\!]_b) & \\ & \Rrightarrow_! & \nu x(Q' \mid !x(b).[\![w]\!]_b) & \\ (i.h.) & \equiv & \nu x(R' \mid !x(b).[\![w]\!]_b) & \\ (i.h.) & \equiv & \nu x([\![u']\!]_a \mid !x(b).[\![w]\!]_b) & =: \quad R \end{array}$$

Note that $[\![u]\!] = [\![u'w]\!] = \lambda\!\!\lambda a.R$. ◀

Summing Up. By iterating the simulation of single steps, and postponing structural equivalence, we obtain our first main result (point 1 is `redn_pi_simulates_redn_mwh` and point 2 is `redn_lwh_simulates_redn_pi` in the sources).

▶ **Theorem 17.**

1. π simulates LSC*: if t is reachable and $t \to^n_{mwh} u$ then for every a there exists Q such that $[\![t]\!]_a \Rrightarrow^n Q$ and $Q \equiv [\![u]\!]_a$.*
2. *LSC simulates π: if $[\![t]\!]_a \Rrightarrow^n Q$ then there exists u such that $t \to^n_{mwh} u$ and $[\![u]\!]_a \equiv Q$.*

By composing Theorem 17 with the equivalence of presentations for the reduction of π (Theorem 12), one can also relate the LSC to the ordinary reduction of π.

7 Relating Weak Head Reduction and Micro Weak Head Reduction

Here we study the relationship between $\to_{wh}$ and $\to_{mwh}$ via the *unfolding* of ESs. We present the pen-and-paper analogous of the Abella formalization, omitting trivial lemmas.

Unfolding. Terms with ES can be *unfolded* into terms without ESs by turning ESs into meta-level substitutions. As explained in Sect. 2, we restrict to *reachable* terms with ESs, characterized by having no ESs in arguments, inside ESs, and under abstractions. Accordingly, the following definition of unfolding $t{\downarrow}$ assumes that it is applied to a reachable term t.

UNFOLDING

$$\begin{array}{rcl@{\qquad}rcl} x{\downarrow} & ::= & x & (tu){\downarrow} & ::= & t{\downarrow}\, u \\ (\lambda x.t){\downarrow} & ::= & \lambda x.t & t[x{\leftarrow}u]{\downarrow} & ::= & t{\downarrow}\,\{x{\leftarrow}u\} \end{array}$$

Projection Via Unfolding. The unfolding turns every weak head β at a distance step $\to_{dB}$ into exactly one weak head step $\to_{wh}$ on the unfolded terms, and every micro substitution step $\to_{ms}$ into an equality. Here we detail only the proof for $\to_{ms}$ steps, which is more interesting and requires an auxiliary lemma.

▶ **Proposition 18.** *Let t be a reachable term. If $t \to_{dB} u$ then $t{\downarrow} \to_{wh} u{\downarrow}$.*

▶ **Lemma 19** (Auxiliary lemma for $\mapsto_{ms}$)**.** *Let $\mathsf{maxw}(t) = W$, t be reachable, u be a term with no ES, and $x \notin \mathsf{fv}(u)$. Then $W\langle x\rangle{\downarrow}\{x{\leftarrow}u\} = W\langle u\rangle{\downarrow}\{x{\leftarrow}u\}$.*

The proof is an easy induction but Abella has trouble with it because the conclusion of the statement is a non-pattern equality. Therefore, the inductive cases need help from the user.

Proof. By induction on $\mathsf{maxw}(t) = C$. Cases:

- $\mathsf{maxw}(y) = \langle\cdot\rangle$, $\mathsf{maxw}(x) = \langle\cdot\rangle$, and $\mathsf{maxw}(\lambda x.r) = \langle\cdot\rangle$ are identical: $W\langle x\rangle{\downarrow}\{x{\leftarrow}u\} = x\{x{\leftarrow}u\} = u = u\{x{\leftarrow}u\} =^* u{\downarrow}\{x{\leftarrow}u\} = W\langle u\rangle{\downarrow}\{x{\leftarrow}u\}$, where the $=^*$ steps holds because unfolding terms with no ES does nothing (it is an easy omitted lemma).

- $\mathsf{maxw}(rw) = W'w$ because $\mathsf{maxw}(r) = W'$. Note that t reachable implies r reachable. By *i.h.*, $W'\langle x\rangle{\downarrow}\{x{\leftarrow}u\} = W'\langle u\rangle{\downarrow}\{x{\leftarrow}u\}$. Then $W\langle x\rangle{\downarrow}\{x{\leftarrow}u\} = W\langle x\rangle{\downarrow}\{x{\leftarrow}u\}w\{x{\leftarrow}u\} = W'\langle u\rangle{\downarrow}\{x{\leftarrow}u\}w\{x{\leftarrow}u\} = W\langle u\rangle{\downarrow}\{x{\leftarrow}u\}$.
- $\mathsf{maxw}(r[y{\leftarrow}w]) = W'[y{\leftarrow}w]$ because $\mathsf{maxw}(r) = W'$. Similar to the previous point. ◀

▶ **Proposition 20.** *Let t be a reachable term.*
1. *If $t \mapsto_{ms} u$ then $t{\downarrow} = u{\downarrow}$.*
2. *If $t \to_{ms} u$ then $t{\downarrow} = u{\downarrow}$.*

Proof.
1. Unfolding the hypothesis, we obtain $t = W\langle x\rangle[x{\leftarrow}r] \mapsto_{ms} W\langle r\rangle[x{\leftarrow}r] = u$. Note that $t{\downarrow} = W\langle x\rangle{\downarrow}\{x{\leftarrow}r\}$ and $u{\downarrow} = W\langle r\rangle{\downarrow}\{x{\leftarrow}r\}$. By Lemma 19, they coincide.
2. By induction on $t \to_{ms} u$ using point 1 in the base case. ◀

Converse Simulation. The converse simulation, that is, that every $\to_{wh}$ step is simulated by a sequence of $\to_{mwh}$ steps, is less easy, and it is where the difficulty of relating small-step and micro-step formalisms lies. While it is true that if t is a term with no ESs and $t \to_{wh} u$ then $t \to_{dB} r$ and $r{\downarrow} = u$, such a property cannot be used for the simulation of rewriting *sequences*, as it does not compose for consecutive steps: if then $u \to_{wh} u'$ we cannot apply the property again because $r \neq u$. One needs the following refined *one-step reflection property*:

$$\text{If } t{\downarrow} \to_{wh} u \text{ then there exists } r \text{ and } w \text{ such that } t \to_{ms}^{*} r \to_{dB} w \text{ and } w{\downarrow} = u$$

To prove such a reflection, we need various properties, in particular that $\to_{ms}$ terminates.

Micro Substitution Normal Terms. For proving the termination of $\to_{ms}$ we use a predicate characterizing $\to_{ms}$-normal terms. We need the concept of answer.

▶ **Definition 21** (Answer). *An* answer *is a term of the form $a ::= \lambda x.t \mid a[x{\leftarrow}t]$.*

The predicate characterizing $\to_{ms}$-normal terms is the disjunction of three predicates.

▶ **Definition 22** (*ms*-normal). *The predicate t is ms-normal is defined as follows.*

$$\frac{a \text{ answer}}{a \text{ is ms-normal}} \qquad \frac{\mathsf{fhv}(t) = x}{t \text{ is ms-normal}} \qquad \frac{t \to_{dB} u}{t \text{ is ms-normal}}$$

The characterization takes the following form in Abella. Curiously, our proof of termination of $\to_{ms}$ relies on the *ms*-normal predicate but never uses its characterization (which we have nonetheless formalized).

▶ **Proposition 23** (Characterization of being $\to_{ms}$-normal). *The following two facts cannot co-exist (that is, together they imply $false$):*
1. *$t \to_{ms} u$;*
2. *t is ms-normal.*

Termination of Micro Substitutions. The proof of termination of $\to_{ms}$ is neat, as we do not need a termination measure, we simply prove it by induction on the structure of terms, via two auxiliary lemmas.

▶ **Lemma 24.** *If W is a weak head context, $W\langle t\rangle$ is ms-normal, and $x \notin \mathsf{fv}(t)$ then $W\langle t\rangle[x{\leftarrow}u]$ is ms-normal.*

Proof. By case analysis of $W\langle t\rangle$ is ms-normal. The answer and $\to_{dB}$-step case are immediate. If $\mathsf{fhv}(W\langle t\rangle) = y \neq x$ then $\mathsf{fhv}(W\langle t\rangle[x\leftarrow u]) = y$, and so $W\langle t\rangle[x\leftarrow u]$ is ms-normal. Finally, $\mathsf{fhv}(W\langle t\rangle) = x$ is impossible, because by hypothesis x does not occur in t. ◀

▶ **Lemma 25.** *If t has no ESs and W is a weak head context not capturing variables of t then $W\langle t\rangle$ is ms-normal.*

Proof. By induction on W. Cases:

- *Empty*, that is, $W = \langle\cdot\rangle$. Then $W\langle t\rangle = t$. The statement is given by the fact that terms with no ESs are ms-normal (easy omitted lemma).
- *Application*, that is $W = W'u$. The statement is given by the *i.h.* and the stability of ms-normal by application (easy omitted lemma).
- *Substitution*, that is $W = W'[x\leftarrow u]$. By *i.h.*, $W'\langle t\rangle$ is ms-normal. By hypothesis, $x \notin \mathsf{fv}(t)$. Then by Lemma 24 $W'\langle t\rangle[x\leftarrow u] = W\langle t\rangle$ is ms-normal. ◀

▶ **Proposition 26** ($\to_{ms}$ terminates)**.** *If t is reachable then $t \to_{ms}^* u$ with u ms-normal.*

Proof. By induction on t is reachable. Cases:

- *Variable* and *abstraction*: immediate since t cannot $\to_{ms}$-reduce and it is ms-normal.
- *Application*, that is $t = rw$: by *i.h.*, $r \to_{ms}^* u'$ with u' ms-normal. Then $rw \to_{ms}^* u'w$ and $u'w$ is ms-normal because being ms-normal is stable by application (omitted lemma).
- *Substitution*, that is $t = r[x\leftarrow w]$: by *i.h.*, $r \to_{ms}^* u'$ with u' ms-normal. Then $r[x\leftarrow w] \to_{ms}^* u'[x\leftarrow w]$. Case analysis of u' ms-normal:
  - If u' is an answer, a $\to_{dB}$-step, or $\mathsf{fhv}(u') = y \neq x$ then so is for $u'[x\leftarrow w]$.
  - $\mathsf{fhv}(u') = x$. Then there is a weak head context W such that $\mathsf{maxw}(u') = W$. Then $u'[x\leftarrow u] \to_{ms} W\langle u\rangle[x\leftarrow u]$. Since W does not capture variables of u, by Lemma 25 we have that $W\langle u\rangle$ is ms-normal. Since $x \notin \mathsf{fv}(u)$, by Lemma 24 $W\langle u\rangle[x\leftarrow u]$ is ms-normal. The statement holds because $r[x\leftarrow w] \to_{ms}^* u'[x\leftarrow w] \to_{ms} W\langle u\rangle[x\leftarrow u]$. ◀

Reflection of $\to_{wh}$ Steps. We can finally prove the one-step reflection property. It rests on the auxiliary case of reflection on ms-normal terms, which is given here without proof.

▶ **Proposition 27** (ms-normal terms reflect $\to_{wh}$ steps as $\to_{dB}$ steps)**.** *If t is reachable, $t{\downarrow} \to_{wh} u$ and t is ms-normal then exists r such that $t \to_{dB} r$.*

▶ **Proposition 28** (Reflection of $\to_{wh}$ steps)**.** *If t is reachable, $t{\downarrow} \to_{wh} u$ then exists r and w such that $t \to_{ms}^* r \to_{dB} w$ and $w{\downarrow} = u$.*

Proof. By termination of $\to_{ms}$ (Prop. 26), there exists r such that $t \to_{ms}^* r$ and r is ms-normal. By $\to_{ms}$-projection (Prop. 20.2), $t{\downarrow} = r{\downarrow}$. We can then apply Prop. 27, obtaining that there exists w such that $r \to_{dB} w$. Since r is reachable, by $\to_{dB}$-projection (Prop. 18) $r{\downarrow} \to_{wh} w{\downarrow}$. Since $\to_{wh}$ is deterministic (Lemma 1), $u = w{\downarrow}$. ◀

Summing Up. We then conclude with our second main result. In the Abella sources, point 1 is `micro_to_small_simulation` and point 2 is `small_to_micro_simulation_no_ES`.

▶ **Theorem 29.** *Let t be reachable.*

1. Micro to small steps*: if $t \to_{mwh}^* u$ then $t{\downarrow} \to_{wh}^* u{\downarrow}$.*
2. Small to micro steps*: if t has no ES and $t \to_{wh}^* u$ then there exists r such that $t \to_{mwh}^* r$ and $r{\downarrow} = u$.*

Proof. Point 1 is an easy induction on the length of $t \to^*_{mwh} u$, using the projection properties (Prop. 18 and Prop. 20). For point 2, one proves that if t is reachable and $t{\downarrow} \to^*_{wh} u$ then there exists r such that $t \to^*_{wh} r$ and $r{\downarrow} = u$, by induction on the length of $t{\downarrow} \to^*_{wh} u$ using the reflection property (Prop. 28). The statement follows from the fact that terms without ES are reachable and satisfy $t{\downarrow} = t$. ◀

Putting it All Together? At this point, it is natural to expect that our two main results – namely the relationships LSC/π (Theorem 17) and λ/LSC (Theorem 29) – can be composed, obtaining a final theorem relating λ and π. This is however not possible, because the key concept for connecting λ and the LSC is the unfolding $t{\downarrow}$ of ESs, which has no analogous on processes. It is exactly such a difficulty that, in presentations without ESs, forces the use of strong bisimulation on processes to close the simulation diagram between λ and π.

At first sight, turning an ES $t[x{\leftarrow}u]$ into a meta-level substitution $t\{x{\leftarrow}u\}$, which is the operation iterated by the unfolding, corresponds on processes to a *broadcast*, that is, to send $[\![u]\!]$ to all sub-processes of $[\![t[x{\leftarrow}u]]\!]_a$ that can perform an input on channel x, but this is misleading because occurrences of x actually correspond to *outputs*, not inputs, and the sub-process encoding $[x{\leftarrow}u]$ is an *input*, not an output.

8 Conclusions

We provide the first formalization of Milner's translation of λ to π, by actually formalizing Accattoli's factorization of the translation via the linear substitution calculus. The difficulties with names and binding are circumvented thanks to the features of the Abella proof assistant, and by defining terms and processes over the same variables.

About future work, it would be interesting to extend our formalization to Sangiorgi's result relating barbed congruence in the π-calculus with his normal form bisimulation in the λ-calculus [41]. It would also be interesting to see how to exploit the new presentation of the rewriting rules of the π-calculus for formalizing other results of its theory.

References

1 Beniamino Accattoli. Proof pearl: Abella formalization of λ-calculus cube property. In Chris Hawblitzel and Dale Miller, editors, *Certified Programs and Proofs - Second International Conference, CPP 2012, Kyoto, Japan, December 13-15, 2012. Proceedings*, volume 7679 of *Lecture Notes in Computer Science*, pages 173–187. Springer, 2012. doi:10.1007/978-3-642-35308-6_15.

2 Beniamino Accattoli. Evaluating functions as processes. In Rachid Echahed and Detlef Plump, editors, *Proceedings 7th International Workshop on Computing with Terms and Graphs, TERMGRAPH 2013, Rome, Italy, 23th March 2013*, volume 110 of *EPTCS*, pages 41–55, 2013. doi:10.4204/EPTCS.110.6.

3 Beniamino Accattoli. Proof nets and the linear substitution calculus. In Bernd Fischer and Tarmo Uustalu, editors, *Theoretical Aspects of Computing - ICTAC 2018 - 15th International Colloquium, Stellenbosch, South Africa, October 16-19, 2018, Proceedings*, volume 11187 of *Lecture Notes in Computer Science*, pages 37–61. Springer, 2018. doi:10.1007/978-3-030-02508-3_3.

4 Beniamino Accattoli, Pablo Barenbaum, and Damiano Mazza. Distilling abstract machines. In Johan Jeuring and Manuel M. T. Chakravarty, editors, *Proceedings of the 19th ACM SIGPLAN international conference on Functional programming, Gothenburg, Sweden, September 1-3, 2014*, pages 363–376. ACM, 2014. doi:10.1145/2628136.2628154.

5 Beniamino Accattoli, Eduardo Bonelli, Delia Kesner, and Carlos Lombardi. A nonstandard standardization theorem. In *The 41st Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL '14, San Diego, CA, USA, January 20-21, 2014*, pages 659–670. ACM, 2014. doi:10.1145/2535838.2535886.

6 Beniamino Accattoli and Ugo Dal Lago. On the invariance of the unitary cost model for head reduction. In Ashish Tiwari, editor, *23rd International Conference on Rewriting Techniques and Applications (RTA'12) , RTA 2012, May 28 - June 2, 2012, Nagoya, Japan*, volume 15 of *LIPIcs*, pages 22–37. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2012. doi:10.4230/LIPIcs.RTA.2012.22.

7 Beniamino Accattoli and Ugo Dal Lago. (leftmost-outermost) beta reduction is invariant, indeed. *Log. Methods Comput. Sci.*, 12(1), 2016. doi:10.2168/LMCS-12(1:4)2016.

8 Beniamino Accattoli, Ugo Dal Lago, and Gabriele Vanoni. The machinery of interaction. In *PPDP '20: 22nd International Symposium on Principles and Practice of Declarative Programming, Bologna, Italy, 9-10 September, 2020*, pages 4:1–4:15. ACM, 2020. doi:10.1145/3414080.3414108.

9 Beniamino Accattoli, Stéphane Graham-Lengrand, and Delia Kesner. Tight typings and split bounds, fully developed. *J. Funct. Program.*, 30:e14, 2020. doi:10.1017/S095679682000012X.

10 Guillaume Ambal, Sergueï Lenglet, and Alan Schmitt. Hoπ in coq. *J. Autom. Reason.*, 65(1):75–124, 2021. doi:10.1007/s10817-020-09553-0.

11 David Baelde, Kaustuv Chaudhuri, Andrew Gacek, Dale Miller, Gopalan Nadathur, Alwen Tiu, and Yuting Wang. Abella: A system for reasoning about relational specifications. *Journal of Formalized Reasoning*, 7(2):1–89, 2014. doi:10.6092/issn.1972-5787/4650.

12 Jesper Bengtson and Joachim Parrow. Formalising the pi-calculus using nominal logic. *Log. Methods Comput. Sci.*, 5(2), 2009. URL: http://arxiv.org/abs/0809.3960.

13 Malgorzata Biernacka, Dariusz Biernacki, Sergueï Lenglet, Piotr Polesiuk, Damien Pous, and Alan Schmitt. Fully abstract encodings of λ-calculus in hocore through abstract machines. In *32nd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2017, Reykjavik, Iceland, June 20-23, 2017*, pages 1–12. IEEE Computer Society, 2017. doi:10.1109/LICS.2017.8005118.

14 Gérard Boudol. The p-calculus in direct style. *High. Order Symb. Comput.*, 11(2):177–208, 1998. doi:10.1023/A:1010064516533.

15 Gérard Boudol and Cosimo Laneve. The discriminating power of multiplicities in the lambda-calculus. *Inf. Comput.*, 126(1):83–102, 1996. doi:10.1006/inco.1996.0037.

16 Xiaojuan Cai and Yuxi Fu. The λ-calculus in the π-calculus. *Math. Struct. Comput. Sci.*, 21(5):943–996, 2011. doi:10.1017/S0960129511000260.

17 Kaustuv Chaudhuri, Matteo Cimini, and Dale Miller. A lightweight formalization of the metatheory of bisimulation-up-to. In Xavier Leroy and Alwen Tiu, editors, *Proceedings of the 4th ACM-SIGPLAN Conference on Certified Programs and Proofs*, pages 157–166, Mumbai, India, January 2015. ACM. doi:10.1145/2676724.2693170.

18 Matteo Cimini, Claudio Sacerdoti Coen, and Davide Sangiorgi. Functions as processes: Termination and the $\lambda\mu\tilde{\mu}$-calculus. In Martin Wirsing, Martin Hofmann, and Axel Rauschmayer, editors, *Trustworthly Global Computing - 5th International Symposium, TGC 2010, Munich, Germany, February 24-26, 2010, Revised Selected Papers*, volume 6084 of *Lecture Notes in Computer Science*, pages 73–86. Springer, 2010. doi:10.1007/978-3-642-15640-3_5.

19 Joëlle Despeyroux. A higher-order specification of the pi-calculus. In *Theoretical Computer Science, Exploring New Frontiers of Theoretical Informatics, International Conference IFIP TCS 2000, Sendai, Japan, August 17-19, 2000, Proceedings*, volume 1872 of *Lecture Notes in Computer Science*, pages 425–439. Springer, 2000. doi:10.1007/3-540-44929-9_30.

20 Adrien Durier, Daniel Hirschkoff, and Davide Sangiorgi. Eager functions as processes. *Theor. Comput. Sci.*, 913:8–42, 2022. doi:10.1016/j.tcs.2022.01.043.

21 Murdoch J. Gabbay. *The π-Calculus in FM*, pages 247–269. Springer Netherlands, Dordrecht, 2003. doi:10.1007/978-94-017-0253-9_10.

22 Andrew Gacek. The abella interactive theorem prover (system description). In Alessandro Armando, Peter Baumgartner, and Gilles Dowek, editors, *Automated Reasoning, 4th International Joint Conference, IJCAR 2008, Sydney, Australia, August 12-15, 2008, Proceedings*, volume 5195 of *Lecture Notes in Computer Science*, pages 154–161. Springer, 2008. doi:10.1007/978-3-540-71070-7_13.

23 Andrew Gacek, Dale Miller, and Gopalan Nadathur. Nominal abstraction. *Inf. Comput.*, 209(1):48–73, 2011. doi:10.1016/j.ic.2010.09.004.

24 Simon J. Gay. A framework for the formalisation of pi calculus type systems in isabelle/hol. In *Theorem Proving in Higher Order Logics, 14th International Conference, TPHOLs 2001, Edinburgh, Scotland, UK, September 3-6, 2001, Proceedings*, volume 2152 of *Lecture Notes in Computer Science*, pages 217–232. Springer, 2001. doi:10.1007/3-540-44755-5_16.

25 Daniel Hirschkoff. A full formalisation of pi-calculus theory in the calculus of constructions. In Elsa L. Gunter and Amy P. Felty, editors, *Theorem Proving in Higher Order Logics, 10th International Conference, TPHOLs'97, Murray Hill, NJ, USA, August 19-22, 1997, Proceedings*, volume 1275 of *Lecture Notes in Computer Science*, pages 153–169. Springer, 1997. doi:10.1007/BFb0028392.

26 Furio Honsell, Marino Miculan, and Ivan Scagnetto. π-calculus in (co)inductive type theories. *Theoretical Computer Science*, 2(253):239–285, 2001. doi:10.1016/S0304-3975(00)00095-5.

27 Guilhem Jaber and Davide Sangiorgi. Games, mobile processes, and functions. In Florin Manea and Alex Simpson, editors, *30th EACSL Annual Conference on Computer Science Logic, CSL 2022, February 14-19, 2022, Göttingen, Germany (Virtual Conference)*, volume 216 of *LIPIcs*, pages 25:1–25:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.CSL.2022.25.

28 Naoki Kobayashi. A partially deadlock-free typed process calculus. *ACM Trans. Program. Lang. Syst.*, 20(2):436–482, 1998. doi:10.1145/276393.278524.

29 Thomas F. Melham. A mechanized theory of the pi-calculus in HOL. *Nord. J. Comput.*, 1(1):50–76, 1994.

30 Dale Miller and Gopalan Nadathur. *Programming with Higher-Order Logic*. Cambridge University Press, 2012. URL: http://www.cambridge.org/de/academic/subjects/computer-science/programming-languages-and-applied-logic/programming-higher-order-logic?format=HB.

31 Dale Miller and Alwen Tiu. A proof theory for generic judgments. *ACM Trans. Comput. Log.*, 6(4):749–783, 2005. doi:10.1145/1094622.1094628.

32 Robin Milner. Functions as processes. *Math. Struct. Comput. Sci.*, 2(2):119–141, 1992. doi:10.1017/S0960129500001407.

33 Robin Milner. Local bigraphs and confluence: Two conjectures: (extended abstract). In Roberto M. Amadio and Iain Phillips, editors, *Proceedings of the 13th International Workshop on Expressiveness in Concurrency, EXPRESS 2006, Bonn, Germany, August 26, 2006*, volume 175 of *Electronic Notes in Theoretical Computer Science*, pages 65–73. Elsevier, 2006. doi:10.1016/j.entcs.2006.07.035.

34 Otmane Aït Mohamed. Mechanizing a pi-calculus equivalence in HOL. In *Higher Order Logic Theorem Proving and Its Applications, 8th International Workshop, Aspen Grove, UT, USA, September 11-14, 1995, Proceedings*, volume 971 of *Lecture Notes in Computer Science*, pages 1–16. Springer, 1995. doi:10.1007/3-540-60275-5_53.

35 Joachim Niehren. Functional computation as concurrent computation. In Hans-Juergen Boehm and Guy L. Steele Jr., editors, *Conference Record of POPL'96: The 23rd ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, Papers Presented at the Symposium, St. Petersburg Beach, Florida, USA, January 21-24, 1996*, pages 333–343. ACM Press, 1996. doi:10.1145/237721.237801.

36 Dominic A. Orchard and Nobuko Yoshida. Using session types as an effect system. In *Proceedings Eighth International Workshop on Programming Language Approaches to Concurrency-*

and Communication-cEntric Software, PLACES 2015, London, UK, 18th April 2015, volume 203 of *EPTCS*, pages 1–13, 2015. doi:10.4204/EPTCS.203.1.

37 Roly Perera and James Cheney. Proof-relevant π-calculus: a constructive account of concurrency and causality. *Math. Struct. Comput. Sci.*, 28(9):1541–1577, 2018. doi:10.1017/S096012951700010X.

38 Christine Röckl, Daniel Hirschkoff, and Stefan Berghofer. Higher-order abstract syntax with induction in isabelle/hol: Formalizing the pi-calculus and mechanizing the theory of contexts. In *Foundations of Software Science and Computation Structures, 4th International Conference, FOSSACS 2001 Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2001 Genova, Italy, April 2-6, 2001, Proceedings*, volume 2030 of *Lecture Notes in Computer Science*, pages 364–378. Springer, 2001. doi:10.1007/3-540-45315-6_24.

39 Davide Sangiorgi. The lazy lambda calculus in a concurrency scenario. *Inf. Comput.*, 111(1):120–153, 1994. doi:10.1006/inco.1994.1042.

40 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.

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

42 Davide Sangiorgi and Xian Xu. Trees from functions as processes. *Log. Methods Comput. Sci.*, 14(3), 2018. doi:10.23638/LMCS-14(3:11)2018.

43 Alwen Tiu and Dale Miller. Proof search specifications of bisimulation and modal logics for the pi-calculus. *ACM Trans. Comput. Log.*, 11(2):13:1–13:35, 2010. doi:10.1145/1656242.1656248.

44 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.

45 Niccolò Veltri and Andrea Vezzosi. Formalizing π-calculus in guarded cubical agda. In Jasmin Blanchette and Catalin Hritcu, editors, *Proceedings of the 9th ACM SIGPLAN International Conference on Certified Programs and Proofs, CPP 2020, New Orleans, LA, USA, January 20-21, 2020*, pages 270–283. ACM, 2020. doi:10.1145/3372885.3373814.