Eager Functions as Processes (long version)

Adrien Durier, Daniel Hirschkoff, Davide Sangiorgi

► **To cite this version:**

Adrien Durier, Daniel Hirschkoff, Davide Sangiorgi. Eager Functions as Processes (long version). Theoretical Computer Science, In press, 10.1016/j.tcs.2022.01.043 . hal-03466150v3

HAL Id: hal-03466150

https://hal.science/hal-03466150v3

Submitted on 4 Feb 2022

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Eager Functions as Processes

Adrien Durier[a], Daniel Hirschkoff[b], Davide Sangiorgi[c]

[a] *École Normale Supérieure de Lyon, 46, allée d'Italie, 69007, Lyon, France*
[b] *École Normale Supérieure de Lyon, 46, allée d'Italie, 69007, Lyon, France*
[c] *Università di Bologna and INRIA, Mura Anteo Zamboni, 7, 40126, Bologna, Italy*

Abstract

We study Milner's encoding of the call-by-value λ-calculus into the π-calculus. We show that, by tuning the encoding to two subcalculi of the π-calculus (Internal π and Asynchronous Local π), the equivalence on λ-terms induced by the encoding coincides with Lassen's eager normal-form bisimilarity, extended to handle η-equality. As behavioural equivalence in the π-calculus we consider contextual equivalence and barbed congruence. We also extend the results to preorders.

A crucial technical ingredient in the proofs is the recently-introduced technique of unique solutions of equations, further developed in this paper. In this respect, the paper also intends to be an extended case study on the applicability and expressiveness of the technique.

Keywords: call-by-value λ-calculus, π-calculus, behavioural equivalence, full-abstraction

Acknowledgements. We are delighted to be able to contribute to the Festschrift in honour of Fu Yuxi. We would like to take this opportunity for heartily thanking him for his many technical and scientific contributions, for his work in favour of the concurrency theory community, both locally in Shanghai and elsewhere, and for all the discussions and time spent together.

Introduction

Milner's work on functions as processes [18, 19], that shows how the evaluation strategies of *call-by-name λ-calculus* and *call-by-value λ-calculus* [1, 23] can be faithfully mimicked in the π-calculus, is generally considered a landmark in Concurrency Theory, and more generally in Programming Language Theory. The comparison with the λ-calculus is a significant expressiveness test for the π-calculus. More than that, it promotes the π-calculus to be a basis for general-purpose programming languages in which communication is the fundamental computing primitive. From the λ-calculus point of view, the comparison provides the means to study λ-terms in contexts other than purely sequential ones, and with the instruments available to reason about processes. Further, Milner's work, and the works that followed it, have contributed to understanding and developing the theory of the π-calculus.

More precisely, Milner shows the operational correspondence between reductions in the λ-terms and in the encoding π-terms. He then uses the correspondence to prove that the encodings are *sound*, i.e., if the processes encoding two λ-terms are behaviourally equivalent, then the source λ-terms are also behaviourally equivalent in the λ-calculus. Milner also shows that the converse, *completeness*, fails, intuitively because the encodings allow one to test the λ-terms in all contexts of the π-calculus — more diverse than those of the λ-calculus.

The main problem that Milner's work left open is the characterisation of the equivalence on λ-terms induced by the encoding, whereby two λ-terms are equal if their encodings are behaviourally equivalent π-calculus terms. The question is largely independent of the precise form of behavioural equivalence adopted in the π-calculus because the encodings are deterministic (or at least confluent). In the paper we consider contextual equivalence (that coincides with may testing and trace equivalence) and barbed congruence (that coincides with bisimilarity).

For the call-by-name λ-calculus, the answer was found shortly later [26, 28]: the equality induced is the equality of Levy-Longo Trees [16], the lazy variant of Böhm Trees. It is actually

[1] Sangiorgi acknowledges support from the MIUR-PRIN project 'Analysis of Program Analyses' (ASPRA, ID: 201784YSZ5_004), and from the European Research Council (ERC) Grant DLV-818616 DIAPASoN.

also possible to obtain Böhm Trees, by modifying the call-by-name encoding so to allow also reductions underneath a λ-abstraction, and by including divergence among the observables [31]. These results show that, at least for call-by-name, the π-calculus encoding, while not fully abstract for the contextual equivalence of the λ-calculus, is in remarkable agreement with the theory of the λ-calculus: several well-known models of the λ-calculus yield Levy-Longo Trees or Böhm Trees as their induced equivalence [15, 16, 4].

For call-by-value, in contrast, the problem of identifying the equivalence induced by the encoding has remained open, for two main reasons. First, tree structures in call-by-value are less studied and less established than in call-by-name. Secondly, proving completeness of an encoding of λ into π requires sophisticated proof techniques. For call-by-name, for instance, a central role is played by *bisimulation up-to contexts*. For call-by-value, however, existing proof techniques, including 'up-to contexts', appeared not to be powerful enough.

In this paper we study the above open problem for call-by-value. Our main result is that the equivalence induced on λ-terms by their call-by-value encoding into the π-calculus is *eager normal-form bisimilarity* [13, 14]. This is a tree structure for call-by-value, proposed by Lassen as the call-by-value counterpart of Levy-Longo Trees. Precisely we obtain the variant that is insensitive to some η-expansions, called *η-eager normal-form bisimilarity*. It validates the η-expansion law for variables:

$$\lambda y.\, xy = x \tag{1}$$

This law is also valid for abstractions: $\lambda y.\,(\lambda z.\, M)y = \lambda z.\, M$ if y does not occur free in M. However, in a weak call-by-value setting, η-expanded terms should not always be equated: indeed, Ω diverges, while $\lambda x.\, \Omega x$ converges to a value.

To obtain the results we have however to make a few adjustments to Milner's encoding and/or specialise the target language of the encoding. These adjustments have to do with the presence of free outputs (outputs of known names) in the encoding. Indeed, Milner had initially translated call-by-value λ-variables using a free output: the translation of the variable x would be a free output $\overline{p}\langle x\rangle$, where p is the continuation, or location, at which x is evaluated. In the original encoding, therefore, the encoding of x at p (the encoding of λ-terms is parametrised upon a name) is defined as $\overline{p}\langle x\rangle$, which can be written as follows:

$$\mathcal{V}[\![x]\!]\langle p\rangle \stackrel{\text{def}}{=} \overline{p}\langle x\rangle. \tag{2}$$

However this is troublesome for the validity of β_{v}-reduction (the property that λ-terms that are related by β_{v}-reduction — the call-by-value β-reduction — are also equal in the π-calculus). Milner solved the problem by ruling out the initial free output $\overline{p}\langle x\rangle$ and by replacing it with a bound output $\boldsymbol{\nu} y\, \overline{p}\langle y\rangle$ followed by a static link $y \blacktriangleright x$. A static link $y \blacktriangleright x$ forwards any name received by y to x, therefore acting as a substitution between x and y (while also constraining the behaviour of the context, that may only use y in output). Thus, in the modified encoding, we have:

$$\mathcal{V}'[\![x]\!]\langle p\rangle \stackrel{\text{def}}{=} \boldsymbol{\nu} y\, (\overline{p}\langle y\rangle.\, y \blacktriangleright x). \tag{3}$$

It was indeed shown later [25] that with (2) the validity of β_{v}-reduction fails. Accordingly, the final journal paper [19] does not even mention encoding (2). If one wants to maintain the simpler rule (2), then the validity of β_{v}-reduction can be regained by taking, as target language, a subset of the π-calculus in which only the output capability of names is communicated. This can be enforced either by imposing a behavioural type system including capabilities [22], or by syntactically taking a dialect of the π-calculus in which only the output capability of names is communicated, such as Local π [17].

The encoding (3) still makes use of free outputs — the final action of $y \blacktriangleright x$ is a free output on x. While this limited form of free output is harmless for the validity of β_{v}-reduction, we show in the paper that this brings problems when analysing λ-terms with free variables: desirable call-by-value equalities fail. An example is given by the law:

$$I(xV) = xV \tag{4}$$

where I is $\lambda z.\, z$ and V is a value.

Law (4) is valid in any model of *call-by-value*, as any context making both terms closed also equates them: we have $\lambda x. I(xV) = \lambda x. xV$, for instance. It also holds in any theory of open call-by-value, as far as we know. This is indeed very natural: for any substitution of x with a closed value, the terms become related by $\beta_{\rm v}$ — and the identity should never be made observable in a theory of the λ-calculus.

Two possible solutions to recover law (4) are:

1. rule out the free outputs; this essentially means transplanting the encoding onto the Internal π-calculus, Iπ [27], a version of the π-calculus in which any name emitted in an output is fresh;

2. control the use of capabilities in the π-calculus; for instance taking Asynchronous Local π, ALπ [17] as the target of the translation. (Controlling capabilities allows one to impose a directionality on names, which, under certain technical conditions, may hide the identity of the emitted names.)

In the paper we consider both approaches, and show that in both cases, the equivalence induced coincides with η-eager normal-form bisimilarity.

In summary, there are two main contributions in the paper:

1. Showing that Milner's encoding fails to equate terms that should be equal in call-by-value.

2. Rectifying the encoding, by considering different target calculi, and investigating Milner's problem in such a setting.

The rectification we make does not really change the essence of the encoding – in one case, the encoding actually remains the same. Moreover, the languages used are well-known dialects of the π-calculus, studied in the literature for other reasons. In the encoding, they allow us to avoid certain accidental misuses of the names emitted in the communications. The calculi were not known at the time of Milner's paper [19].

A key role in the completeness proof is played by the technique of *unique solution of equations*, proposed in [7]. The structure induced by Milner's call-by-value encoding was expected to look like Lassen's trees; however existing proof techniques did not seem powerful enough to prove it. The unique solution technique allows one to derive process bisimilarities from equations whose infinite unfolding does not introduce divergences, by proving that the processes are solutions of the same equations. The technique can be generalised to possibly-infinite systems of equations, and can be strengthened by allowing certain kinds of divergences in equations. In this respect, another goal of the paper is to carry out an extended case study on the applicability and expressiveness of the techniques. Then, a by-product of the study are a few further developments of the technique. In particular, one such result allows us to transplant uniqueness of solutions from a system of equations, for which divergences are easy to analyse, to another one. Another result is about the application of the technique to preorders.

Finally, we consider preorders — thus referring to the preorder on λ-terms induced by a behavioural preorder on their π-calculus encodings. We introduce a preorder on Lassen's trees (preorders had not been considered by Lassen) and show that this is the preorder on λ-terms induced by the call-by-value encoding, when the behavioural relation on π-calculus terms is the ordinary contextual preorder (again, with the same restrictions as mentioned above). With the move from equivalences to preorders, the overall structure of the proofs of our full abstraction results remains the same. However, the impact on the application of the unique-solution technique is substantial, because the phrasing of this technique in the cases of preorders and of equivalences is quite different.

Further related work. The standard behavioural equivalence in the λ-calculus is contextual equivalence. Encodings into the π-calculus (be it for call-by-name or call-by-value) break contextual equivalence because π-calculus contexts are richer than those in the (pure) λ-calculus. In the paper we try to understand how far beyond contextual equivalence the discriminating power of the π-calculus brings us, for call-by-value. The opposite approach is to restrict the set of 'legal' π-contexts so to remain faithful to contextual equivalence. This approach has been followed, for call-by-name, and using type systems, in [5, 33].

Open call-by-value has been studied in [3], where the focus is on operational properties of λ-terms; behavioural equivalences are not considered. An important difference with our work is that in [3], $xV_1 \cdots V_k$ is treated as a value, i.e., β-reduction can be triggered when the argument has this shape.

An extensive presentation of call-by-value, including denotational models, is Ronchi della Rocca and Paolini's book [24].

In [7], the unique-solution technique is used in the completeness proof for Milner's call-by-name encoding. That proof essentially revisits the proof of [28], which is based on bisimulation up-to context. We have explained above that the case for call-by-value is quite different.

Structure of the paper. We recall basic definitions about the call-by-value λ-calculus and the π-calculus in Section 1. The technique of unique solution of equations is introduced in Section 2, together with some new developments. Section 3 presents our analysis of Milner's encoding, beginning with the shortcomings related to the presence of free outputs. The first solution to these shortcomings is to move to the Internal π-calculus: this is described in Section 4. For the proof of completeness, in Section 4.4, we rely on unique solution of equations; we also compare such technique with the 'up-to techniques'. The second solution is to move to the Asynchronous Local π-calculus: this is discussed in Section 5. We show in Section 6 how our results can be adapted to preorders and to contextual equivalence. Finally in Section 7 we present conclusions and directions for future work.

Comparison with the results published in [8]. This paper is an extended version of [8]. We provide here detailed proofs which were either absent or only sketched in [8], notably for: the soundness of the encoding into Iπ (Sections 4.2 and 4.3, Appendix C), completeness of the same encoding (Section 4.4), the unique solution technique for contextual relations and preorders (Section 6.2, Appendix E), as well as some more details about the full abstraction proofs for contextual preorders (Section 6.3) and the encoding into ALπ (Section 5.2). We also include more detailed discussions along the paper, notably about Milner's encoding (Section 3), the encoding into Iπ (Section 4.1), and the encoding into ALπ (Section 5.1).

1. Background material

Throughout the paper, $\mathcal{R}$ ranges over relations. The composition of two relations $\mathcal{R}$ and $\mathcal{R}'$ is written $\mathcal{R}\,\mathcal{R}'$. We often use infix notation for relations; thus $P \mathrel{\mathcal{R}} Q$ means $(P, Q) \in \mathcal{R}$. A tilde represents a tuple. The i-th element of a tuple $\widetilde{P}$ is referred to as P_i. Our notations are extended to tuples componentwise. Thus $\widetilde{P} \mathrel{\mathcal{R}} \widetilde{Q}$ means $P_i \mathrel{\mathcal{R}} Q_i$ for all components. Several behavioural relations are used in this paper — Appendix A presents a summary of these.

1.1. The call-by-value λ-calculus

We let x and y range over the set of λ-calculus variables. The set Λ of λ-terms is defined by the grammar

$$M ::= x \;\Big|\; \lambda x.\, M \;\Big|\; M_1 M_2 \,.$$

Free variables, closed terms, substitution, α-conversion etc. are defined as usual [4, 9]. Here and in the rest of the paper (including when reasoning about π processes), we adopt the usual "Barendregt convention". This will allow us to assume freshness of bound variables and names whenever needed. The set of free variables in the term M is written $\mathsf{fv}(M)$, and we sometimes use $\mathsf{fv}(M, N)$ to denote $\mathsf{fv}(M) \cup \mathsf{fv}(N)$. Application is left-associative; therefore MNL is $(MN)L$. We abbreviate $\lambda x_1. \cdots . \lambda x_n.\, M$ as $\lambda x_1 \cdots x_n.\, M$, or $\lambda \widetilde{x}.\, M$ if the length of $\widetilde{x}$ is not important. Symbol Ω stands for the always-divergent term $(\lambda x.\, xx)(\lambda x.\, xx)$.

A *context* is a term with a hole $[\cdot]$, possibly occurring more than once. If C is a context, then $C[M]$ is a shorthand for C where the hole $[\cdot]$ is substituted by M. An *evaluation context*, ranged over using C_{e}, is a special kind of inductively defined context, with exactly one hole $[\cdot]$, and in which a term replacing the hole can immediately run. In the pure λ-calculus *values* are abstractions and variables.

$$\begin{array}{rrcl} \text{Evaluation contexts} & C_{\mathrm{e}} & := & [\cdot] \mid C_{\mathrm{e}} M \mid V C_{\mathrm{e}} \\ \text{Values} & V & := & x \mid \lambda x.\, M \end{array}$$

We accordingly write $\mathsf{fv}(C_e)$ for the free variables of C_e.

Eager reduction (or β_v-reduction), $\longrightarrow \subseteq \Lambda \times \Lambda$, is defined on open terms, and is determined by the rule:

$$C_e[(\lambda x.\, M)V] \longrightarrow C_e[M\{V/x\}],$$

where $\{V/x\}$ stands for the capture-avoiding substitution of x with V.

A term in *eager normal form* is a term that has no eager reduction. We write $\Longrightarrow$ for the reflexive transitive closure of $\longrightarrow$.

Proposition 1. *The following hold:*

1. *If $M \longrightarrow M'$, then $C_e[M] \longrightarrow C_e[M']$ and $M\sigma \longrightarrow M'\sigma$, for any substitution σ that replaces variables with values.*
2. *Terms in eager normal form are either values or admit a unique decomposition of the shape $C_e[xV]$.*

Therefore, given a term M, either $M \Longrightarrow M'$ where M' is a term in eager normal form, or there is an infinite reduction sequence starting from M. In the first case, we say that M *has eager normal form* M', written $M \Downarrow M'$; in the second M *diverges*, written $M \Uparrow$. We write $M \Downarrow$ when $M \Downarrow M'$ for some M'.

Definition 2 (Contextual equivalence)**.** Given $M,\ N \in \Lambda$, we say that M and N are contextually equivalent, written $M \simeq^{\Lambda}_{\mathrm{ct}} N$, if for any context C, we have $C[M] \Downarrow$ iff $C[N] \Downarrow$.

1.2. Tree Semantics for call-by-value

In this section, we recall Lassen's *eager normal-form bisimilarity* [13, 14, 32].

Definition 3 (Eager normal-form bisimulation)**.** A relation $\mathcal{R}$ between λ-terms is an *eager normal-form bisimulation* if, whenever $M \mathrel{\mathcal{R}} N$, one of the following holds:

1. both M and N diverge;
2. $M \Downarrow C_e[xV]$ and $N \Downarrow C'_e[xV']$ for some x, values V, V', and evaluation contexts C_e and C'_e satisfying $V \mathrel{\mathcal{R}} V'$ and $C_e[z] \mathrel{\mathcal{R}} C'_e[z]$ for a fresh z;
3. $M \Downarrow \lambda x.\, M'$ and $N \Downarrow \lambda x.\, N'$ for some x, M', N' with $M' \mathrel{\mathcal{R}} N'$;
4. $M \Downarrow x$ and $N \Downarrow x$ for some x.

Eager normal-form bisimilarity, written $\Leftrightarrow$, is the largest eager normal-form bisimulation.

Essentially, the structure of a λ-term that is unveiled by Definition 3 is that of a (possibly infinite) tree obtained by repeatedly applying β_v-reduction, and branching a tree whenever instantiation of a variable is needed to continue the reduction (clause (2)). We call such trees *Eager Trees* (ETs); accordingly, we also call eager normal-form bisimilarity the *Eager-Tree equality.*

Example 4. Relation $\Leftrightarrow$ is strictly finer than contextual equivalence $\simeq^{\Lambda}_{\mathrm{ct}}$: the inclusion $\Leftrightarrow \subseteq \simeq^{\Lambda}_{\mathrm{ct}}$ follows from the congruence properties of $\Leftrightarrow$ [13]. For strictness, examples are given by the following equalities, which hold for $\simeq^{\Lambda}_{\mathrm{ct}}$ but not for $\Leftrightarrow$:

$$\Omega = (\lambda y.\, \Omega)(xV) \qquad\qquad xV = (\lambda y.\, xV)(xV)\ .$$

Example 5 (η rule)**.** The η-rule is not valid for $\Leftrightarrow$. For instance, we have $\Omega \not\Leftrightarrow \lambda x.\, \Omega x$. The rule is not even valid on values, as we also have $\lambda y.\, xy \not\Leftrightarrow x$. It holds however for abstractions: $\lambda y.\, (\lambda x.\, M)\ y \Leftrightarrow \lambda x.\, M$ when $y \notin \mathsf{fv}(M)$.

The failure of the η-rule $\lambda y.\, xy \not\Leftrightarrow x$ is troublesome as, under any closed value substitution (a substitution replacing variables with closed values), the two terms are indeed eager normal-form bisimilar. Thus *η-eager normal-form bisimilarity* [13] takes η-expansion into account so to recover such missing equalities.

Definition 6 (η-eager normal-form bisimulation)**.** A relation $\mathcal{R}$ between λ-terms is an *η-eager normal-form bisimulation* if, whenever $M \mathrel{\mathcal{R}} N$, either one of the clauses of Definition 3, or one of the two following additional clauses, hold:

5. $M \Downarrow x$ and $N \Downarrow \lambda y.\, N'$ for some x, y, and N' such that $N' \Downarrow C_{\mathrm{e}}[xV]$, with $y \mathrel{\mathcal{R}} V$ and $z \mathrel{\mathcal{R}} C_{\mathrm{e}}[z]$ for some value V, evaluation context C_{e}, and fresh z.

6. the converse of (5), i.e., $N \Downarrow x$ and $M \Downarrow \lambda y.\, M'$ for some x, y, and M' such that $M' \Downarrow C_{\mathrm{e}}[xV]$, with $V \mathrel{\mathcal{R}} y$ and $C_{\mathrm{e}}[z] \mathrel{\mathcal{R}} z$ for some value V, evaluation context C_{e}, and fresh z.

Then *η-eager normal-form bisimilarity*, $\Leftrightarrow_\eta$, is the largest η-eager normal-form bisimulation.

We sometimes call relation $\Leftrightarrow_\eta$ the *η-Eager-Tree equality.*

Remark 7. Definition 6 coinductively allows η-expansions to occur underneath other η-expansions, hence trees with infinite η-expansions may be equated with finite trees. For instance, we have

$$x \Leftrightarrow_\eta \lambda y.\, xy \Leftrightarrow_\eta \lambda y.\, x(\lambda z.\, yz) \Leftrightarrow_\eta \lambda y.\, x(\lambda z.\, y(\lambda w.\, zw)) \Leftrightarrow_\eta \ldots$$

An example of a finite tree being equated with an infinite tree by $\Leftrightarrow_\eta$ is as follows: take a fixpoint combinator Y, and define $f \stackrel{\text{def}}{=} (\lambda zxy.\, x(z\, y))$. We then have $Y f x \Longrightarrow \lambda y.\, x(Y\, f\, y)$, and then $x\,(Y\, f\, y) \Longrightarrow x(\lambda z.\, y\,(Y\, f\, z))$, and so on. Hence, we have $x \Leftrightarrow_\eta Y f x$.

1.3. The π-calculus

In all encodings we consider, the encoding of a λ-term is parametric on a name, i.e., it is a function from names to π-calculus processes. We also need parametric processes (over one or several names) to write recursive process definitions and equations. We call such parametric processes *abstractions.* The instantiation of the parameters of an abstraction F is done via the *application* construct $F\langle\widetilde{a}\rangle$. We use P, Q for processes, F for abstractions. Processes and abstractions form the set of *π-agents* (or simply *agents*), ranged over by A. Small letters $a, b, \ldots, x, y, \ldots$ range over the infinite set of names. The grammar of the π-calculus is thus:

$$\begin{array}{lcll}
A & ::= & P \;\big|\; F & \text{(agents)} \\
P & ::= & \mathbf{0} \;\big|\; a(\widetilde{b}).\, P \;\big|\; \overline{a}\langle\widetilde{b}\rangle.\, P \;\big|\; \boldsymbol{\nu} a\, P & \text{(processes)} \\
 & & \;\big|\; P_1 \mid P_2 \;\big|\; !a(\widetilde{b}).\, P \;\big|\; F\langle\widetilde{a}\rangle & \\
F & ::= & (\widetilde{a})\; P \;\big|\; K & \text{(abstractions)}
\end{array}$$

$\mathbf{0}$ is the inactive process. An input-prefixed process $a(\widetilde{b}).\, P$, where $\widetilde{b}$ has pairwise distinct components, waits for a tuple of names $\widetilde{c}$ to be sent along a and then behaves like $P\{\widetilde{c}/\widetilde{b}\}$, where $\{\widetilde{c}/\widetilde{b}\}$ is the simultaneous substitution of names $\widetilde{b}$ with names $\widetilde{c}$ (see below). An output particle $\overline{a}\langle\widetilde{b}\rangle$ emits names $\widetilde{b}$ at a. Parallel composition is used to run two processes in parallel. The restriction $\boldsymbol{\nu} a\, P$ makes name a local, or private, to P. A replicated input $!a(\widetilde{b}).\, P$ stands for a countable infinite number of copies of $a(\widetilde{b}).\, P$ in parallel. (Replication could be avoided in the syntax since it can be encoded with recursion. However its semantics is simple, and it is a useful construct for examples and encodings; thus we chose to include it in the grammar.)

We do not include the operators of sum and matching. We assign parallel composition the lowest precedence among the operators. We refer to [20] for detailed discussions on the operators of the language.

We use α to range over prefixes. In prefixes $a(\widetilde{b})$ and $\overline{a}\langle\widetilde{b}\rangle$, we call a the *subject* and $\widetilde{b}$ the *object.* When the tilde is empty, the surrounding brackets in prefixes will be omitted. We often abbreviate $\alpha.\, \mathbf{0}$ as α, and $\boldsymbol{\nu} a\, \boldsymbol{\nu} b\, P$ as $(\boldsymbol{\nu} a, b)P$. An input prefix $a(\widetilde{b}).\, P$, a restriction $\boldsymbol{\nu} b\, P$, and an abstraction $(\widetilde{b})\; P$ are binders for names $\widetilde{b}$ and b, respectively, and give rise in the expected way to the definition of *free names* (fn) and *bound names* (bn) of a term or a prefix. An agent is *name-closed* if it does not contain free names. (Since the number of recursive definitions may be infinite, some care is necessary in the definition of free names of an agent, using a least fixed-point construction.) As in the λ-calculus, we identify processes or actions which only differ in the choice of the bound names. The symbol $=$ will mean "syntactic identity modulo α-conversion".

Sometimes, we use $\stackrel{\text{def}}{=}$ as abbreviation mechanism, to assign a name to an expression to which we want to refer later.

We use constants, ranged over by K, for writing recursive definitions. Each constant has a defining equation of the form $K \triangleq (\widetilde{x})\ P$, where $(\widetilde{x})\ P$ is name-closed; $\widetilde{x}$ are the formal parameters of the constant (replaced by the actual parameters whenever the constant is used).

Since the calculus is polyadic, we assume a *sorting system* [20] to avoid disagreements in the arities of the tuples of names carried by a given name and in applications of abstractions. We do not present the sorting system because it is not essential. The reader should take for granted that all agents described obey a sorting.

A *context* C of the π-calculus is a π-agent in which some subterms have been replaced by the hole $[\cdot]$ or, if the context is polyadic, with indexed holes $[\cdot]_1, \dots, [\cdot]_n$. Then, $C[A]$ or $C[\widetilde{A}]$ is the agent resulting from replacing the holes with the terms A or $\widetilde{A}$. Holes in contexts have a sort too, as they could be in place of an abstraction.

Substitutions are of the form $\{\widetilde{b}/\widetilde{a}\}$, and are finite assignments of names to names. We use σ and ρ to range over substitutions. The application of a substitution σ to an expression H is written $H\sigma$. Substitutions have precedence over the operators of the language; $\sigma\rho$ is the composition of substitutions where σ is performed first, therefore $P\sigma\rho$ is $(P\sigma)\rho$.

The Barendregt convention allows us to assume that the application of a substitution does not affect bound names of expressions; similarly, when comparing the transitions of two processes, we assume that the bound names of the actions do not occur free in the processes. In a statement, we say that a name is *fresh* to mean that it is different from any other name which occurs in the statement or in objects of the statement like processes and substitutions.

Abstraction and application. We say that an application redex $((\widetilde{x})P)\langle\widetilde{a}\rangle$ can be *normalised* as $P\{\widetilde{a}/\widetilde{x}\}$. An agent is *normalised* if all such application redexes have been contracted, everywhere in the terms. When reasoning on behaviours it is useful to assume that all expressions are normalised, in the above sense. Thus in the remainder of the paper *we identify an agent with its normalised expression*. The application construct $F\langle\widetilde{a}\rangle$ will play an important role in the treatment of equations in the following sections.

1.4. Operational semantics

Transitions of π-calculus processes are of the form $P \xrightarrow{\mu} P'$, where the grammar for actions is given by

$$\mu \quad ::= \quad a(\widetilde{b}) \;\Big|\; \boldsymbol{\nu}\widetilde{d}\,\overline{a}\langle\widetilde{b}\rangle \;\Big|\; \tau\ .$$

- $P \xrightarrow{a(\widetilde{b})} P'$ is an input, where $\widetilde{b}$ are the names bound by the input prefix which is being fired (we adopt a late version of the Labelled Transition Semantics),
- $P \xrightarrow{\boldsymbol{\nu}\widetilde{d}\,\overline{a}\langle\widetilde{b}\rangle} P'$ is an output, where $\widetilde{d} \subseteq \widetilde{b}$ are private names extruded in the output, and
- $P \xrightarrow{\tau} P'$ is an internal action.

We abbreviate $\boldsymbol{\nu}\widetilde{d}\,\overline{a}\langle\widetilde{b}\rangle$ as $\overline{a}\langle\widetilde{b}\rangle$ when $\widetilde{d}$ is empty. The occurrences of $\widetilde{b}$ in $a(\widetilde{b})$ and those of $\widetilde{d}$ in $\boldsymbol{\nu}\widetilde{d}\,\overline{a}\langle\widetilde{b}\rangle$ are bound; we define accordingly the sets of bound names and free names of an action μ, respectively written $\mathrm{bn}(\mu)$ and $\mathrm{fn}(\mu)$. The set of all the names appearing in μ (both free and bound) is written $\mathrm{n}(\mu)$.

Figure 1 presents the transition rules for the π-calculus.

We write $\Rightarrow$ for the reflexive transitive closure of $\xrightarrow{\tau}$, and $\stackrel{\mu}{\Longrightarrow}$ for $\Longrightarrow\xrightarrow{\mu}\Longrightarrow$. Then $\stackrel{\hat{\mu}}{\Longrightarrow}$ (resp. $\xrightarrow{\hat{\mu}}$) is $\stackrel{\mu}{\Longrightarrow}$ (resp. $\xrightarrow{\mu}$) if μ is not τ, and $\Rightarrow$ (resp. $\xrightarrow{\tau}$ or $=$) otherwise. In bisimilarity or other behavioural relations for the π-calculus we consider, no name instantiation is used in the input clause or elsewhere; technically, the relations are *ground*. In the subcalculi we consider ground bisimilarity is a congruence and coincides with barbed congruence (congruence breaks in the full π-calculus). Besides the simplicity of their definition, the ground relations make more effective the theory of unique solutions of equations (in particular, checking divergences is simpler, see Section 2).

The reference behavioural equivalence for π-calculi is the usual *barbed congruence*. We recall its definition, on a generic subset $\mathcal{L}$ of π-calculus processes. A $\mathcal{L}$-*context* is a process of $\mathcal{L}$ with

$$\frac{}{a(\widetilde{b}).\, P \xrightarrow{a(\widetilde{b})} P} \qquad \frac{}{!a(\widetilde{b}).\, P \xrightarrow{a(\widetilde{b})} !a(\widetilde{b}).\, P \mid P} \qquad \frac{}{\overline{a}\langle\widetilde{b}\rangle.\, P \xrightarrow{\overline{a}\langle\widetilde{b}\rangle} P}$$

$$\frac{P \xrightarrow{\boldsymbol{\nu}\widetilde{d}\,\overline{a}\langle\widetilde{b}\rangle} P'}{\boldsymbol{\nu}n\, P \xrightarrow{\boldsymbol{\nu}(\{n\}\cup\widetilde{d})\,\overline{a}\langle\widetilde{b}\rangle} P'}\ \begin{array}{l} n \in \widetilde{b} \\ n \notin \widetilde{d} \end{array} \qquad \frac{P \xrightarrow{\mu} P'}{\boldsymbol{\nu}n\, P \xrightarrow{\mu} \boldsymbol{\nu}n\, P'}\, n \notin \mathrm{n}(\mu) \qquad \frac{P \xrightarrow{a(\widetilde{b})} P' \qquad Q \xrightarrow{\boldsymbol{\nu}\widetilde{d}\,\overline{a}\langle\widetilde{b'}\rangle} Q'}{P \mid Q \xrightarrow{\tau} \boldsymbol{\nu}\widetilde{d}\,(P'\{\widetilde{b'}/\widetilde{b}\} \mid Q')}$$

$$\frac{P \xrightarrow{\mu} P'}{P \mid Q \xrightarrow{\mu} P' \mid Q}\,\mathrm{bn}(\mu) \cap \mathrm{fn}(Q) = \emptyset \qquad \frac{P\{\widetilde{b}/\widetilde{a}\} \xrightarrow{\mu} P'}{((\widetilde{a})\ P)\langle\widetilde{b}\rangle \xrightarrow{\mu} P'} \qquad \frac{F\langle\widetilde{a}\rangle \xrightarrow{\mu} P'}{K\langle\widetilde{a}\rangle \xrightarrow{\mu} P'}\ \text{if } K \triangleq F$$

Figure 1: Labelled Transition Semantics for the π-calculus

a single hole $[\cdot]$ in it (the hole has a sort, as it could be in place of an abstraction). We write $P \Downarrow_a$ if P can make an output action whose subject is a, possibly after some internal moves.

We make only output observable because this is standard in asynchronous calculi; in the case of a synchronous calculus like Iπ, Definition 8 below yields synchronous barbed congruence, and adding also observability of inputs does not change the induced equivalence. More details on this are given in Section 5.

Definition 8 (Barbed congruence)**.** *Barbed bisimilarity* is the largest symmetric relation $\dot{\simeq}$ on π-calculus processes such that $P \dot{\simeq} Q$ implies:

1. If $P \Longrightarrow P'$ then there is Q' such that $Q \Longrightarrow Q'$ and $P' \dot{\simeq} Q'$.

2. $P \Downarrow_a$ iff $Q \Downarrow_a$.

Let $\mathcal{L}$ be a set of π-calculus agents, and $A, B \in \mathcal{L}$. We say that A and B are *barbed congruent in* $\mathcal{L}$, written $A \simeq^{\mathcal{L}} B$, if for each (well-sorted) $\mathcal{L}$-context C, it holds that $C[A] \dot{\simeq} C[B]$.

Remark 9. We have defined barbed congruence uniformly on processes and abstractions (via a quantification on all process contexts). Usually, however, definitions will only be given for processes; it is then intended that they are extended to abstractions by requiring closure under ground parameters, i.e., by supplying fresh names as arguments.

As for all contextually-defined behavioural relations, so barbed congruence is hard to work with. In all calculi we consider, it can be characterised in terms of *ground bisimilarity*, under the (mild) condition that the processes are image-finite up to weak bisimilarity. (We recall that the class of processes *image-finite up to weak bisimilarity* is the largest subset $\mathcal{IF}$ of π-calculus processes which is closed by transitions and such that $P \in \mathcal{IF}$ implies that, for all actions μ, the set $\{P' \mid P \stackrel{\mu}{\Longrightarrow} P'\}$ quotiented by weak bisimilarity is finite. The definition is extended to abstractions as by Remark 9.) All the agents in the paper, including those obtained by encodings of the λ-calculus, are image-finite up to weak bisimilarity. The distinctive feature of *ground* bisimilarity is that it does not involve instantiation of the bound names of inputs (other than by means of fresh names), and similarly for abstractions. In the remainder, we omit the adjective 'ground'.

Definition 10 (Bisimilarity)**.** A symmetric relation $\mathcal{R}$ on π-processes is a *bisimulation*, if whenever $P \mathrel{\mathcal{R}} Q$ and $P \xrightarrow{\mu} P'$, then $Q \stackrel{\hat{\mu}}{\Longrightarrow} Q'$ for some Q' with $P' \mathrel{\mathcal{R}} Q'$.

Processes P and Q are *bisimilar*, written $P \approx Q$, if $P \mathrel{\mathcal{R}} Q$ for some bisimulation $\mathcal{R}$.

We extend $\approx$ to abstractions, as per Remark 9: $F \approx G$ if $F\langle\widetilde{b}\rangle \approx G\langle\widetilde{b}\rangle$ for fresh $\widetilde{b}$.

In the proofs, we shall also use *strong bisimilarity*, written $\sim$. Relation $\sim$ is defined as per Definition 10, but Q must answer with a strong transition, that is, we impose $Q \xrightarrow{\mu} Q'$.

The Expansion preorder. We define the expansion preorder, written $\preceq$, where $P \preceq Q$ intuitively means that P and Q have the same behaviour, and that P may not be 'slower' (in the sense of doing more $\xrightarrow{\tau}$ transitions) than process Q.

Definition 11. *Expansion*, written $\preceq$, is defined as the largest relation $\mathcal{R}$ such that $P \mathrel{\mathcal{R}} Q$ implies

1. if $P \xrightarrow{\mu} P'$, then for some Q', $Q \stackrel{\mu}{\Longrightarrow} Q'$ and $P' \mathrel{\mathcal{R}} Q'$, and
2. if $Q \xrightarrow{\mu} Q'$, then for some P', $P \xrightarrow{\hat{\mu}} P'$ and $P' \mathrel{\mathcal{R}} Q'$.

The converse of $\preceq$ is written $\succeq$.

As usual, expansion is extended to abstractions by requiring ground instantiation of the parameters: $F \preceq F'$ if $F\langle\widetilde{a}\rangle \approx F'\langle\widetilde{a}\rangle$, where $\widetilde{a}$ are fresh names of the appropriate sort.

In the following we shall use two standard properties of expansion: first, expansion is finer than bisimilarity, i.e., $\preceq \subseteq \approx$. Second, expansion, like bisimilarity, is preserved by all contexts.

1.5. The Subcalculi Iπ and ALπ

We focus on two subcalculi of the π-calculus: the Internal π-calculus (Iπ), and the Asynchronous Local π-calculus (ALπ). They are obtained by placing certain constraints on prefixes.

Iπ. In Iπ, all outputs are bound. This is syntactically enforced by replacing the output construct with the bound-output construct $\overline{a}(\widetilde{b}).\,P$, which, with respect to the grammar of the ordinary π-calculus, is an abbreviation for $\boldsymbol{\nu}\widetilde{b}\,\overline{a}\langle\widetilde{b}\rangle.\,P$. In all tuples (input, output, abstractions, applications) the components are pairwise distinct so to make sure that distinctions among names are preserved by reduction.

Theorem 12. *In Iπ, on agents that are image-finite up to $\approx$, barbed congruence and bisimilarity coincide.*

The encoding of the λ-calculus into Iπ yields processes that are image-finite up to $\approx$. Thus we can use bisimilarity as a proof technique for barbed congruence.

ALπ. ALπ is defined by enforcing that in an input $a(\widetilde{b}).\,P$, all names in $\widetilde{b}$ appear only in output position in P. Moreover, ALπ being *asynchronous*, output prefixes have no continuation; in the grammar of the π-calculus this corresponds to having only outputs of the form $\overline{a}\langle\widetilde{b}\rangle.\,\mathbf{0}$ (which we will simply write $\overline{a}\langle\widetilde{b}\rangle$).

In ALπ, to maintain the characterisation of barbed congruence as (ground) bisimilarity, the transition system has to be modified [17], allowing the dynamic introduction of additional processes (the 'links', sometimes also called forwarders). In Section 5, we present the modified transition system for ALπ, upon which weak bisimilarity is defined. We also explain how they allow us to obtain for ALπ a property similar to that of Theorem 12 for Iπ.

2. Unique solutions in Iπ and ALπ

We adapt the proof technique of unique solution of equations, from [7] to the calculi Iπ and ALπ, in order to derive bisimilarity results. The technique is discussed in [7] on the asynchronous π-calculus (for possibly-infinite systems of equations). The structure of the proofs for Iπ and ALπ is similar; in particular the completeness part is essentially the same because bisimilarity is the same. The differences in the syntax of Iπ, and in the transition system of ALπ, show up only in certain technical details of the soundness proofs.

The results presented in this section hold both for Iπ and for ALπ.

Equation expressions. We need variables to write equations. We use capital letters X, Y, Z for these variables and call them *equation variables* (sometimes simply *variables*). The body of an equation is a name-closed abstraction possibly containing equation variables (that is, applications can also be of the form $X\langle\widetilde{a}\rangle$). Thus, the solutions of equations are abstractions. Free names of equation expressions and contexts are defined as for agents.

We use E to range over name-closed abstractions; and $\mathcal{E}$ to range over systems of equations, defined as follows. In all the definitions, the indexing set I can be infinite.

Definition 13. Assume that, for each i belonging to a countable indexing set I, we have a variable X_i, and an expression E_i, possibly containing some variables. Then $\{X_i = E_i\}_{i\in I}$ (sometimes written $\widetilde{X} = \widetilde{E}$) is a *system of equations*. (There is one equation for each variable X_i; we sometimes use X_i to refer to that equation.)

A system of equations is *guarded* if each occurrence of a variable in the body of an equation is underneath a prefix.

$E[\widetilde{F}]$ is the abstraction resulting from E by replacing each occurrence of the variable X_i with the abstraction F_i (as usual assuming $\widetilde{F}$ and $\widetilde{X}$ have the same sort). This is syntactic replacement. However recall that we identify an agent with its normalised expression: hence replacing X with $(\widetilde{x})P$ in $X\langle\widetilde{a}\rangle$ is the same as replacing $X\langle\widetilde{a}\rangle$ with the process $P\{\widetilde{a}/\widetilde{x}\}$.

Definition 14. Suppose $\{X_i = E_i\}_{i\in I}$ is a system of equations. We say that:

- $\widetilde{F}$ is a *solution of the system of equations for* $\approx$ if for each i it holds that $F_i \approx E_i[\widetilde{F}]$.
- The system has *a unique solution for* $\approx$ if whenever $\widetilde{F}$ and $\widetilde{G}$ are both solutions for $\approx$, we have $\widetilde{F} \approx \widetilde{G}$.

Definition 15 (Syntactic solutions)**.** The *syntactic solutions* of the system of equations $\widetilde{X} = \widetilde{E}$ are the recursively defined constants $\mathrm{K}_{\widetilde{E},i} \triangleq E_i[\widetilde{\mathrm{K}}_{\widetilde{E}}]$, for each $i \in I$, where I is the indexing set of the system of equations.

The syntactic solutions of a system of equations are indeed solutions of it.

A process P *diverges* if it can perform an infinite sequence of internal moves, possibly after some visible ones (i.e., actions different from τ); formally, there are processes P_i, $i \geq 0$, and some n, such that $P = P_0 \xrightarrow{\mu_0} P_1 \xrightarrow{\mu_1} P_2 \xrightarrow{\mu_2} \dots$ and for all $i > n$, $\mu_i = \tau$. We call a *divergence of* P the sequence of transitions $\big(P_i \xrightarrow{\mu_i} P_{i+1}\big)_i$. In the case of an abstraction F, as per Remark 9, F has a divergence if the process $F\langle\widetilde{a}\rangle$ has a divergence, where $\widetilde{a}$ are fresh names. A tuple of agents $\widetilde{A}$ *is divergence-free* if none of the components A_i has a divergence.

The following result is the technique we rely on to establish completeness of the encoding. As announced above, it holds both in Iπ and in ALπ.

Theorem 16. *In Iπ and ALπ, a guarded system of equations with divergence-free syntactic solutions has unique solution for $\approx$.*

The proof of Theorem 16 is very similar to a similar theorem, for the π-calculus, which is presented in [7]. Moreover, we present a proof of unique solution for trace inclusion and trace equivalence (in Iπ) in Appendix E. The latter is also very similar in structure to the two aforementioned proofs. For a pedagogical presentation of these proofs, we refer the reader to [7], specifically to the proof of unique solution for weak bisimilarity in the setting of CCS.

Techniques for ensuring termination, hence divergence freedom, have been studied in, e.g., [34, 6, 29].

Further Developments of the Technique of Unique Solution of Equations.

We present some further developments to the theory of unique solution of equations, that are needed for the results in this paper. The first result allows us to derive the unique-solution property for a system of equations from the analogous property of an extended system.

Definition 17. A system of equations $\mathcal{E}'$ *extends* system $\mathcal{E}$ if there exists a fixed set of indices J such that any solution of $\mathcal{E}$ can be obtained from a solution of $\mathcal{E}'$ by removing the components corresponding to indices in J.

Lemma 18. *Consider two systems of equations $\mathcal{E}'$ and $\mathcal{E}$ where $\mathcal{E}'$ extends $\mathcal{E}$. If $\mathcal{E}'$ has a unique solution, then the property also holds for $\mathcal{E}$.*

We shall use Lemma 18 in Section 4.4, in a situation where we transform a certain system into another one, whose uniqueness of solutions is easier to establish. Then, by Lemma 18, the property holds for the initial system.

Remark 19. We cannot derive Lemma 18 by comparing the syntactic solutions of the two systems $\mathcal{E}'$ and $\mathcal{E}$. For instance, the equations $X = \tau.X$ and $X = \tau.\tau\ldots$ (where $\tau.\tau\ldots$ abbreviates the corresponding recursive definition) have strongly bisimilar syntactic solutions, yet only the latter equation has the unique-solution property. (Further, Lemma 18 allows us to compare systems of different size).

The second development is a generalisation of Theorem 16 and Lemma 18 to trace equivalence and to trace preorders; we postpone its presentation to Section 6.

3. Milner's Encodings

3.1. Background: encodings $\mathcal{V}$ and $\mathcal{V}'$

Milner noticed [18, 19] that his call-by-value encoding can be easily tuned so to mimic forms of evaluation in which, in an application MN, the function M is run first, or the argument N is run first, or function and argument are run in parallel (the proofs are actually carried out for this last option). We chose here the first option.

The core of any encoding of the λ-calculus into a process calculus is the translation of function application. This becomes a particular form of parallel combination of two processes, the function and its argument; β_{v}-reduction is then modelled as process interaction.

The encoding of a λ-term is parametric over a name; this may be thought of as the *location* of that term, or as its *continuation*. A term that becomes a value signals so at its continuation name and, in doing so, it grants access to the body of the value. Such body is replicated, so that the value may be copied several times. When the value is a function, its body can receive two names: (the access to) its value argument, and the following continuation. In the translation of application, first the function is run, then the argument; finally the function is informed of its argument and continuation.

In the original paper [18], Milner presented two candidates for the encoding of call-by-value λ-calculus [23]. They follow the same pattern of translation, in particular regarding application (as described above), but with a technical difference in the rule for variables. One encoding, $\mathcal{V}$, is defined as follows (for the case of application, we adapt the encoding from parallel call-by-value to left-to-right call by value, as described above):

$$\begin{aligned}
\mathcal{V}[\![\lambda x.\, M]\!] &\stackrel{\mathrm{def}}{=} (p)\ \overline{p}(y).\, !y(x,q).\, \mathcal{V}[\![M]\!]\langle q\rangle \\
\mathcal{V}[\![MN]\!] &\stackrel{\mathrm{def}}{=} (p)\ (\boldsymbol{\nu} q\ )(\mathcal{V}[\![M]\!]\langle q\rangle \mid q(y).\, \boldsymbol{\nu} r\ (\mathcal{V}[\![N]\!]\langle r\rangle \mid r(w).\, \overline{y}\langle w,p\rangle)) \\
\mathcal{V}[\![x]\!] &\stackrel{\mathrm{def}}{=} (p)\ \overline{p}\langle x\rangle
\end{aligned}$$

In the other encoding, $\mathcal{V}'$, application and λ-abstraction are treated as in $\mathcal{V}$; the rule for variables is:

$$\mathcal{V}'[\![x]\!] \stackrel{\mathrm{def}}{=} (p)\ \overline{p}(y).\, !y(z,q).\, \overline{x}\langle z,q\rangle\ \ .$$

In $\mathcal{V}'$, a λ-calculus variable gives rise to a one-place buffer. As the computation proceeds, these buffers are chained together, gradually increasing the number of steps necessary to simulate a β-reduction. This phenomenon does not occur in $\mathcal{V}$, where the occurrence of a variable disappears after it is used. Hence the encoding $\mathcal{V}$ is more efficient than $\mathcal{V}'$,

3.2. Some problems with the encodings

The immediate free output in the encoding of variables in $\mathcal{V}$ breaks the validity of β_{v}-reduction; i.e., there exist a term M and a value V such that $\mathcal{V}[\![(\lambda x.\, M)V]\!] \not\approx \mathcal{V}[\![M\{V/x\}]\!]$ [25]. The encoding $\mathcal{V}'$ fixes this by communicating, instead of a free name, a fresh pointer to that name. Technically, the initial free output of x is replaced by a bound output coupled with a link to x (the process $!y(z,q).\, \overline{x}\langle z,q\rangle$, receiving at y and re-emitting at x). Thus β_{v}-reduction is validated, i.e., $\mathcal{V}[\![(\lambda x.\, M)V]\!] \approx \mathcal{V}[\![M\{V/x\}]\!]$ for any M and V [25].

(The final version of Milner's paper [19] was written after the results in [25] were known and presents only the encoding $\mathcal{V}'$.)

Nevertheless, $\mathcal{V}'$ only delays the free output, as the added link contains itself a free output. As a consequence, we can show that other desirable equalities of call-by-value are broken in $\mathcal{V}'$. An example is law (4) from the Introduction, as stated by Proposition 20 below. This law is desirable (and indeed valid for contextual equivalence, or the Eager-Tree equality) intuitively because, in any substitution closure of the law, either both terms diverge, or they converge to the same value. The same argument holds for their λ-closures, $\lambda x.\, xV$ and $\lambda x.\, I(xV)$.

We recall that $\simeq^{\pi}$ is barbed congruence in the π-calculus.

Proposition 20 (Non-law)**.** *For any value V, we have:*

$$\mathcal{V}[\![I(xV)]\!] \not\simeq^{\pi} \mathcal{V}[\![xV]\!] \quad \textit{and} \quad \mathcal{V}'[\![I(xV)]\!] \not\simeq^{\pi} \mathcal{V}'[\![xV]\!] \ .$$

(The law is violated also under coarser equivalences, such as contextual equivalence.) Technically, the reason why the law fails in π can be illustrated when $V = y$, for encoding $\mathcal{V}$. We have:

$$\mathcal{V}[\![xy]\!]\langle p\rangle \simeq^{\pi} \overline{x}\langle y, p\rangle$$
$$\mathcal{V}[\![I(xy)]\!]\langle p\rangle \simeq^{\pi} (\boldsymbol{\nu} q\,)(\overline{x}\langle y, q\rangle \mid q(z).\, \overline{p}\langle z\rangle)$$

(details of the calculation may be found in Appendix B)

In presence of the normal form xy, the identity I becomes observable. Indeed, in the second term, a fresh name, q, is sent instead of continuation p, and a link between q and p is installed. This corresponds to a law which is valid in ALπ, but not in π.

Remark 21 (Generalisations of law (4))**.** The phenomenon observed in law 4 can be observed in a more general setting. We can observe that for any evaluation context C_{e} and any value V, we have

$$\mathcal{V}[\![C_{\mathrm{e}}[I(xV)]]\!]\langle p\rangle \not\simeq^{\pi} \mathcal{V}[\![C_{\mathrm{e}}[xV]]\!]\langle p\rangle \ .$$

One may want to generalise further this law, by replacing the identity I by an arbitrary function $\lambda z.\, M$, provided M somehow "uses" z. We may take M to be equal to $C'_{\mathrm{e}}[z]$, for some evaluation context C'_{e}, or any term that reduces to such a normal form. We can then show, indeed:

$$\mathcal{V}[\![C_{\mathrm{e}}[\lambda z.\, M(xV)]]\!]\langle p\rangle \not\simeq^{\pi} \mathcal{V}[\![C_{\mathrm{e}}[M\{xV/z\}]]\!]\langle p\rangle \ .$$

Generalising further, to a term M whose normal form is not written $C'_{\mathrm{e}}[z]$, is outside the scope of this work. This could be related to Accattoli and Sacerdoti Coen's notion of "useful reduction" for the call-by-value λ-calculus [2].

3.3. Well-behaved encodings

The problem put forward in Proposition 20 can be avoided by iterating the transformation that takes us from $\mathcal{V}$ to $\mathcal{V}'$ (i.e., the replacement of a free output with a bound output so to avoid all emissions of free names). Thus the target language becomes Internal π; the resulting encoding is analysed in Section 4.

Another solution is to control the use of name capabilities in processes. In this case the target language becomes ALπ, and we need not modify the initial encoding $\mathcal{V}$. This situation is analysed in Section 5.

In both solutions, the encoding uses two kinds of names: *value names* $x, y, \dots, v, w, \dots$ and *continuation names* $p, q, r, \dots,$. For simplicity, we assume that the set of value names contains the set of λ-variables.

Continuation names are always used linearly, meaning that they are only used once in subject position (once in input and once in output). On the other hand, value names may be used multiple times. Continuation names are used to transmit a value name, and value names are used to transmit a pair consisting of a value name and a continuation name. This is a very mild form of typing. We could avoid the distinction between these two kinds of names, at the cost of introducing additional replications in the encoding.

Moreover, in both solutions, the use of link processes validates the following law — a form of η-expansion — (the law fails for Milner's encoding into the π-calculus):

$$\lambda y.\, xy = x \ .$$

In the call-by-value λ-calculus this is a useful law, that holds because substitutions replace variables with values.

$$\mathcal{I}[\![\lambda x.\, M]\!] \overset{\text{def}}{=} (p)\ \overline{p}(y).\, !y(x,q).\, \mathcal{I}[\![M]\!]\langle q\rangle$$
$$\mathcal{I}[\![x]\!] \overset{\text{def}}{=} (p)\ \overline{p}(y).\, y \triangleright x$$
$$\mathcal{I}[\![MN]\!] \overset{\text{def}}{=} (p)\ \boldsymbol{\nu} q\ \big(\mathcal{I}[\![M]\!]\langle q\rangle \mid q(y).\, \boldsymbol{\nu} r\ \big(\mathcal{I}[\![N]\!]\langle r\rangle \mid r(w).\, \overline{y}(w',p').\, (w' \triangleright w \mid p' \triangleright p)\big)\big)$$

Figure 2: The encoding into Iπ

4. Encoding in the Internal π-calculus

We present the encoding in Section 4.1, together with an optimised version of it, which is actually the main object of study to establish soundness. We then proceed to establish validity of β_{v}-reduction and operational correspondence, both for the optimised encoding, in Section 4.2. This allows us to derive soundness w.r.t. $\leftrightarrows_\eta$ of the original (unoptimised) encoding in Section 4.3. In Section 4.4, we establish the completeness of the encoding, using the unique solution technique.

Some proofs are omitted from the main text, and are given in Appendix C.

4.1. The Encoding and its Optimised Version

Figure 2 presents the encoding into Iπ, derived from Milner's encoding by removing the free outputs as explained in Section 3. Process $a \triangleright b$ represents a *link* (sometimes called forwarder; for readability we use the infix notation $a \triangleright b$ for the constant $\triangleright$). It transforms all outputs at a into outputs at b (therefore a, b are names of the same sort). The body of $a \triangleright b$ is replicated, unless a and b are *continuation names*. The definition of the constant $\triangleright$ therefore is:

$$\triangleright \ \triangleq \ \begin{cases} (p,q)\ p(x).\, \overline{q}(y).\, y \triangleright x & \text{if } p,q \text{ are continuation names} \\ (x,y)\ !x(z,p).\, \overline{y}(w,q).\, (q \triangleright p \mid w \triangleright z) & \text{if } x,y \text{ are value names} \end{cases}$$

(The distinction between continuation names and value names is not necessary, but simplifies the proofs.)

We recall some useful properties related to compositions of links [27].

Lemma 22. *We have:*

1. $\boldsymbol{\nu} q\ (p \triangleright q \mid q \triangleright r) \succeq p \triangleright r$*, for all continuation names* p, r*.*
2. $\boldsymbol{\nu} y\ (x \triangleright y \mid y \triangleright z) \succeq x \triangleright z$*, for all value names* x, z*.*

The Optimised Encoding. In order to establish operational correspondence (Proposition 27) we introduce an *optimised encoding*: we remove some of the internal transitions from the encoding of Figure 2, as they prevent the use of the expansion $\succeq$. This allows us to guarantee that if $M \Longrightarrow N$, the encoding of N is faster than the encoding of M, in the sense that it performs fewer internal steps before a visible transition. As a consequence, the encoding of a term in normal form is ready to perform a visible transition. The results about the optimised encoding are formulated using the expansion preorder, which is useful for the soundness proof. For the completeness proof, we use these results with $\approx$ in place of $\succeq$.

We therefore relate λ-terms and Iπ-terms via the optimised encoding $\mathcal{O}$, presented in Figure 3. In the figure we assume that rules VAR-VAL and ABS-VAL have priority over the others; accordingly, in rules VAL-APP, APP-VAL, and APP, terms M and N should not be values.

The optimised encoding is obtained from that in Figure 2 by performing a few (deterministic) reductions, at the price of a more complex definition. Precisely, in the encoding of application we remove some of the initial communications, including those with which a term signals to have become a value. To achieve this, the encoding of an application is split into several cases, depending on whether a subterm of the application is a value or not. This yields to a case distinction according to whether the components in an application are values or not. This is close in spirit to the idea of the *colon translation* in [23].

$$
\begin{array}{rcll}
\mathcal{O}[\![xV]\!] & \stackrel{\text{def}}{=} & (p)\ \overline{x}(z,q).\,(\mathcal{O}_{\mathrm{V}}[\![V]\!]\langle z\rangle \mid q \triangleright p) & \text{VAR-VAL} \\
\mathcal{O}[\![(\lambda x.\,M)V]\!] & \stackrel{\text{def}}{=} & (p)\ \boldsymbol{\nu} y,w\,(\mathcal{O}_{\mathrm{V}}[\![\lambda x.\,M]\!]\langle y\rangle \mid \mathcal{O}_{\mathrm{V}}[\![V]\!]\langle w\rangle & \text{ABS-VAL} \\
 & & \quad \mid \overline{y}(w',r').\,(w' \triangleright w \mid r' \triangleright p)) & \\
\mathcal{O}[\![VM]\!] & \stackrel{\text{def}}{=} & (p)\ \boldsymbol{\nu} y\,(\mathcal{O}_{\mathrm{V}}[\![V]\!]\langle y\rangle \mid \boldsymbol{\nu} r\,(\mathcal{O}[\![M]\!]\langle r\rangle & \text{VAL-APP} \\
 & & \quad \mid r(w).\,\overline{y}(w',r').\,(w' \triangleright w \mid r' \triangleright p))) & \\
\mathcal{O}[\![MV]\!] & \stackrel{\text{def}}{=} & (p)\ \boldsymbol{\nu} q\,(\mathcal{O}[\![M]\!]\langle q\rangle \mid q(y).\,\boldsymbol{\nu} w\,(\mathcal{O}_{\mathrm{V}}[\![V]\!]\langle w\rangle & \text{APP-VAL} \\
 & & \quad \mid \overline{y}(w',r').\,(w' \triangleright w \mid r' \triangleright p))) & \\
\mathcal{O}[\![MN]\!] & \stackrel{\text{def}}{=} & (p)\ \boldsymbol{\nu} q\,(\mathcal{O}[\![M]\!]\langle q\rangle \mid q(y).\,\boldsymbol{\nu} r\,(\mathcal{O}[\![N]\!]\langle r\rangle \mid & \text{APP} \\
 & & \quad r(w).\,\overline{y}(w',r').\,(w' \triangleright w \mid r' \triangleright p))) & \\
\mathcal{O}[\![V]\!] & \stackrel{\text{def}}{=} & (p)\ \overline{p}(y).\,\mathcal{O}_{\mathrm{V}}[\![V]\!]\langle y\rangle & \text{OPT-VAL}
\end{array}
$$

where $\mathcal{O}_{\mathrm{V}}[\![V]\!]$ is thus defined:

$$
\begin{array}{rcll}
\mathcal{O}_{\mathrm{V}}[\![\lambda x.\,M]\!] & \stackrel{\text{def}}{=} & (y)\ !y(x,q).\,\mathcal{O}[\![M]\!]\langle q\rangle & \text{OPT-ABS} \\
\mathcal{O}_{\mathrm{V}}[\![x]\!] & \stackrel{\text{def}}{=} & (y)\ y \triangleright x & \text{OPT-VAR}
\end{array}
$$

Moreover, in rules VAL-APP and APP-VAL, M is not a value; in rule APP M and N are not values.

Figure 3: Optimised encoding into Iπ

The general idea of the optimised encoding can be illustrated on two particular cases. For $\mathcal{O}[\![VM]\!]$, the corresponding equation is the result of unfolding the original encoding, and performing one (deterministic) communication. In the case of $\mathcal{O}[\![xV]\!]$, not only do we unfold the original encoding and reduce along deterministic communications, but we also perform the administrative reductions that always precede the execution of $\mathcal{I}[\![xV]\!]$.

The next lemma builds on Lemma 22 to show that, on the processes obtained by the encoding into Iπ, links behave as substitutions. We recall that p, q are continuation names, whereas x, y are value names.

Lemma 23. *We have:*

1. $\boldsymbol{\nu} x\,(\mathcal{O}[\![M]\!]\langle p\rangle \mid x \triangleright y) \succeq \mathcal{O}[\![M\{y/x\}]\!]\langle p\rangle$.
2. $\boldsymbol{\nu} p\,(\mathcal{O}[\![M]\!]\langle p\rangle \mid p \triangleright q) \succeq \mathcal{O}[\![M]\!]\langle q\rangle$.
3. $\boldsymbol{\nu} y\,(\mathcal{O}_{\mathrm{V}}[\![V]\!]\langle y\rangle \mid x \triangleright y) \succeq \mathcal{O}_{\mathrm{V}}[\![V]\!]\langle x\rangle$.

The following lemma, relating the original and the optimised encoding, allows us to use the latter to establish the soundness of the former. The proofs of Lemmas 23 and 24 are presented in Appendix C.2.

Lemma 24. $\mathcal{I}[\![M]\!] \succeq \mathcal{O}[\![M]\!]$*, for all* $M \in \Lambda$.

4.2. Operational Correspondence

Thanks to the optimised encoding, we can now formulate and prove the operational correspondence between the (optimised) encoding and the source λ-terms.

The proofs of the two following lemmas are presented in Appendix C.2.

We first establish that reduction in the λ-calculus yields expansion for the encodings of the λ-terms.

Lemma 25 (Validity of β_{v}-reduction)**.** *For any* M, N *in* Λ*,* $M \longrightarrow N$ *implies that for any* p*,* $\mathcal{O}[\![M]\!]\langle p\rangle \succeq \mathcal{O}[\![N]\!]\langle p\rangle$.

To prove operational correspondence, we need a technical lemma about the optimised encoding of terms in eager normal form that are not values.

Lemma 26. *We have:*

$$\mathcal{O}[\![C_{\mathrm{e}}[xV]]\!]\langle p\rangle \sim \overline{x}(z,q).\,(\mathcal{O}_{\mathrm{V}}[\![V]\!]\langle z\rangle \mid q(y).\,\mathcal{O}[\![C_{\mathrm{e}}[y]]\!]\langle p\rangle).$$

In the lemma below, recall that we identify processes or transitions that only differ in the choice of the bound names. Therefore, when we say a process has exactly one immediate transition, we mean that there is a unique pair (μ,P), up to alpha-conversion, such that $\mathcal{O}[\![M]\!]\langle p\rangle \xrightarrow{\mu} P$.

Proposition 27 (Operational correspondence)**.** *For any $M \in \Lambda$ and fresh p, process $\mathcal{O}[\![M]\!]\langle p\rangle$ has exactly one immediate transition, and exactly one of the following clauses holds:*

1. $\mathcal{O}[\![M]\!]\langle p\rangle \xrightarrow{\overline{p}(y)} P$ *and M is a value, with $P = \mathcal{O}_{\mathrm{V}}[\![M]\!]\langle y\rangle$;*
2. $\mathcal{O}[\![M]\!]\langle p\rangle \xrightarrow{\overline{x}(z,q)} P$ *and $M = C_{\mathrm{e}}[xV]$ and $P \succeq \mathcal{O}_{\mathrm{V}}[\![V]\!]\langle z\rangle \mid q(y).\,\mathcal{O}[\![C_{\mathrm{e}}[y]]\!]\langle p\rangle$;*
3. $\mathcal{O}[\![M]\!]\langle p\rangle \xrightarrow{\tau} P$ *and there is N with $M \longrightarrow N$ and $P \succeq \mathcal{O}[\![N]\!]\langle p\rangle$.*

Proof. We rely on Proposition 1 to distinguish two cases:

- M is in eager normal form.

  Then (i) either M is a value, and we are in the first case above, by definition; or (ii) we are in the second case above, and we rely on Lemma 26 to conclude.

- There exists N such that $M \longrightarrow N$, We then use the property that is established in the proof of validity of β_{v}-reduction (Lemma 25), namely that $\mathcal{O}[\![M]\!]\langle p\rangle \xrightarrow{\tau}\succeq \mathcal{O}[\![N]\!]\langle p\rangle$.

□

The operational correspondence has two immediate consequences, regarding converging and diverging terms.

Lemma 28. *If $\mathcal{O}[\![M]\!]\langle p\rangle \stackrel{\mu}{\Longrightarrow} P$ and $\mu \neq \tau$, then M admits an eager normal form M' such that $\mathcal{O}[\![M']\!]\langle p\rangle \xrightarrow{\mu} P_0$ and $P \succeq P_0$ for some P_0.*

Proof. By induction on the length of the reduction $\mathcal{O}[\![M]\!]\langle p\rangle \stackrel{\mu}{\Longrightarrow} P$. If $\mathcal{O}[\![M]\!]\langle p\rangle \xrightarrow{\mu} P$ and $\mu \neq \tau$, by Proposition 27, M is an eager normal form. Otherwise, there is P' such that $\mathcal{O}[\![M]\!]\langle p\rangle \xrightarrow{\tau} P' \stackrel{\mu}{\Longrightarrow} P$; by Proposition 27, there is N such that $M \longrightarrow N$ and $P' \succeq \mathcal{O}[\![N]\!]\langle p\rangle$. Therefore, since $P' \stackrel{\mu}{\Longrightarrow} P$, there is Q such that $\mathcal{O}[\![N]\!]\langle p\rangle \stackrel{\mu}{\Longrightarrow} Q$, $P \succeq Q$ and, by definition of expansion, the sequence of transitions from $\mathcal{O}[\![N]\!]\langle p\rangle$ to Q is shorter than the sequence from P' to P. By the induction hypothesis, N admits an eager normal form N', with $\mathcal{O}[\![N']\!]\langle p\rangle \xrightarrow{\mu} Q_0$ and $Q \succeq Q_0$ for some Q_0.

N' being an eager normal form for N, it is also an eager normal form for M. Moreover, we deduce from $Q \succeq Q_0$ and $P \succeq Q$ that $P \succeq Q_0$. □

Lemma 29. *$\mathcal{O}[\![M]\!]\langle p\rangle \approx \mathbf{0}$ iff $M \Uparrow$.*

Proof.
1. Suppose $\mathcal{O}[\![M]\!] \not\approx (p)\ \mathbf{0}$. Then $M \xrightarrow{\mu} P$ for some $\mu \neq \tau$, and by Lemma 28, M has an eager normal form (hence M does not diverge).
2. Assume now $\mathcal{O}[\![M]\!] \approx (p)\ \mathbf{0}$. By Proposition 27, $\mathcal{O}[\![M]\!]\langle p\rangle \xrightarrow{\tau} P$ for some P, and there is N such that $M \longrightarrow N$, and $\mathcal{O}[\![N]\!]\langle p\rangle \approx P$. Since $\mathcal{O}[\![M]\!] \approx (p)\ \mathbf{0}$, we have $P \approx \mathbf{0}$, thus $\mathcal{O}[\![N]\!] \approx (p)\ \mathbf{0}$. With this property, we can construct an infinite sequence of reductions from M, thus $M \Uparrow$.

□

4.3. Soundness

The structure of the proof of soundness of the encoding is similar to that for the analogous property for Milner's call-by-name encoding with respect to Levy-Longo Trees [28]. The details are however different, as in call-by-value both the encoding and the trees (the Eager Trees extended to handle η-expansion) are more complex.

Using the operational correspondence, we then show that the observables for bisimilarity in the encoding π-terms imply the observables for η-eager normal-form bisimilarity in the encoded λ-terms. The delicate cases are those in which a branch in the tree of the terms is produced — case (2) of Definition 3 — and where an η-expansion occurs — thus a variable is equivalent to an abstraction, cases (5) and (6) of Definition 6.

For the branching, we exploit a decomposition property on π-terms, roughly allowing us to derive from the bisimilarity of two parallel compositions the componentwise bisimilarity of the single components. For the η-expansion, if $\mathcal{I}[\![x]\!] \approx \mathcal{I}[\![\lambda z. M]\!]$, where $M \Downarrow C_{\mathrm{e}}[xV]$, we use a coinductive argument to derive $V \Leftrightarrow_\eta z$ and $C_{\mathrm{e}}[y] \Leftrightarrow_\eta y$, for y fresh; from this we then obtain $\lambda z. M \Leftrightarrow_\eta x$.

The following lemma allows us to decompose an equivalence between two parallel processes. This result is used to handle equalities of the form $\mathcal{I}[\![C_{\mathrm{e}}[xV]]\!] \approx \mathcal{I}[\![C'_{\mathrm{e}}[xV']]\!]$, in order to deduce equivalence between V and V' on the one hand, and between $C_{\mathrm{e}}[y]$ and $C'_{\mathrm{e}}[y]$ on the other.

Lemma 30. *Suppose that a does not occur free in Q or Q', and one of the following holds:*

1. $a(x).\, P \mid Q \approx a(x).\, P' \mid Q'$;
2. $!a(x).\, P \mid Q \approx\, !a(x).\, P' \mid Q'$.

Then we also have $Q \approx Q'$.

Lemma 31. *Suppose $!y(z,q).\, \mathcal{O}[\![M]\!]\langle q\rangle \approx\, !y(z,q).\, \mathcal{O}[\![N]\!]\langle q\rangle$, where y does not occur in M, N. Then $\mathcal{O}[\![M]\!]\langle q\rangle \approx \mathcal{O}[\![N]\!]\langle q\rangle$.*

Proof. We first observe that if $\mathcal{O}[\![N]\!]\langle p\rangle \Rightarrow P$, then there exists N' such that $N \longrightarrow^* N'$, $P \succeq \mathcal{O}[\![N']\!]\langle p\rangle$ and $\mathcal{O}[\![N]\!]\langle p\rangle \approx P$. This follows from operational correspondence (Proposition 27) and validity of β_{v}-reduction (Lemma 25).

Let us now prove the lemma. We play a transition $\xrightarrow{y(z,q)}$ on the left hand side. The answer on the right hand side leads to a process P such that $\mathcal{O}[\![N]\!]\langle q\rangle \Rightarrow P$ and

$$!y(z,q).\, \mathcal{O}[\![M]\!]\langle q\rangle \mid \mathcal{O}[\![M]\!]\langle q\rangle \approx\, !y(z,q).\, \mathcal{O}[\![N]\!]\langle q\rangle \mid P.$$

By the observation above, we deduce that

$$!y(z,q).\, \mathcal{O}[\![M]\!]\langle q\rangle \mid \mathcal{O}[\![M]\!]\langle q\rangle \approx\, !y(z,q).\, \mathcal{O}[\![N]\!]\langle q\rangle \mid \mathcal{O}[\![N]\!]\langle q\rangle.$$

We can then conclude using Lemma 30. □

We now show that the only λ-terms whose encoding is bisimilar to the encoding of some variable x reduce either to x, or to a (possibly infinite) η-expansion of x.

Lemma 32. *If V is a value and x a variable, $\mathcal{O}_{\mathrm{V}}[\![V]\!] \approx \mathcal{O}_{\mathrm{V}}[\![x]\!]$ implies that either $V = x$ or $V = \lambda z. M$, where the eager normal form of M is of the form $C_{\mathrm{e}}[xV']$, with $\mathcal{O}[\![V']\!] \approx \mathcal{O}[\![z]\!]$ and $\mathcal{O}[\![C_{\mathrm{e}}[y]]\!] \approx \mathcal{O}[\![y]\!]$ for any y fresh.*

Proof. We observe that if $y \neq x$, then $\mathcal{O}_{\mathrm{V}}[\![y]\!] \not\approx \mathcal{O}_{\mathrm{V}}[\![x]\!]$; therefore if $\mathcal{O}_{\mathrm{V}}[\![V]\!] \approx \mathcal{O}_{\mathrm{V}}[\![x]\!]$ and $V \neq x$, then V has to be an abstraction $V = \lambda z. M$. In this case, by definition, we have:

- $\mathcal{O}[\![\lambda z. M]\!]\langle p\rangle = \overline{p}(y).\, !y(z,q).\, \mathcal{O}[\![M]\!]\langle q\rangle$ and
- $\mathcal{O}[\![x]\!]\langle p\rangle = \overline{p}(y).\, !y(z,q).\, \overline{x}(z',q').\, (z' \triangleright z \mid q' \triangleright q)$.

Therefore $\mathcal{O}[\![M]\!]\langle q\rangle \approx \overline{x}(z',q').\, (z' \triangleright z \mid q' \triangleright q)$, and by Lemma 28 and Proposition 27, M has an eager normal form $C_{\mathrm{e}}[xV']$. We have, using Lemma 26 (since $\succeq \subseteq \approx$):

$$\begin{array}{rcl} \mathcal{O}[\![M]\!]\langle q\rangle & \approx & \mathcal{O}[\![C_{\mathrm{e}}[xV']]\!]\langle q\rangle \\ & \approx & \overline{x}(z',q').\, (\mathcal{O}_{\mathrm{V}}[\![V']\!]\langle z'\rangle \mid q'(y).\, \mathcal{O}[\![C_{\mathrm{e}}[y]]\!]\langle q\rangle), \end{array}$$

which gives

$$\mathcal{O}_{\mathrm{V}}[\![V']\!]\langle z'\rangle \mid q'(y).\, \mathcal{O}[\![C_{\mathrm{e}}[y]]\!]\langle q\rangle \approx z' \triangleright z \mid q' \triangleright q.$$

We observe that z' does not occur free in $q'(y).\, \mathcal{O}[\![C_{\mathrm{e}}[y]]\!]\langle q\rangle$ and q' does not occur free in $\mathcal{O}_{\mathrm{V}}[\![V']\!]\langle z'\rangle$. Furthermore, $\mathcal{O}_{\mathrm{V}}[\![V']\!]\langle z'\rangle$ is prefixed by an input on z'. By applying Lemma 30 twice, we deduce

$$\mathcal{O}_{\mathrm{V}}[\![V']\!]\langle z'\rangle \approx z' \triangleright z \quad \text{and} \quad q'(y).\, \mathcal{O}[\![E[y]]\!]\langle q\rangle \approx q' \triangleright q\ .$$

By definition, $z' \triangleright z = \mathcal{O}_{\mathrm{V}}[\![z]\!]\langle z'\rangle$, so we have $\mathcal{O}[\![V']\!]\langle z'\rangle \approx \mathcal{O}[\![z]\!]\langle z'\rangle$.

We also have

$$q'(y).\,\mathcal{O}[\![C_{\mathrm{e}}[y]]\!]\langle q\rangle \approx q' \triangleright q = q'(y).\,\overline{q}(y').\,(y' \triangleright y),$$

which gives

$$\mathcal{O}[\![C_{\mathrm{e}}[y]]\!]\langle q\rangle \approx \overline{q}(y').\,(y' \triangleright y) = \mathcal{O}[\![y]\!]\langle q\rangle.$$

□

We can now prove soundness of the encoding.

Proposition 33 (Soundness)**.** *For any $M, N \in \Lambda$, if $\mathcal{I}[\![M]\!] \approx \mathcal{I}[\![N]\!]$ then $M \Leftrightarrow_\eta N$.*

Proof. Let $\mathcal{R} \stackrel{\text{def}}{=} \{(M, N) \mid \mathcal{O}[\![M]\!] \approx \mathcal{O}[\![N]\!]\}$; we show that $\mathcal{R}$ is an η-eager normal-form bisimulation, and conclude by Lemma 24. Assume $\mathcal{O}[\![M]\!] \approx \mathcal{O}[\![N]\!]$.

1. If $M \Uparrow$, by Lemma 29, for any fresh p, $\mathcal{O}[\![M]\!]\langle p\rangle \approx \mathbf{0}$. Thus $\mathcal{O}[\![N]\!]\langle p\rangle \approx \mathbf{0}$, and, by Lemma 29 again, $N \Uparrow$.

2. Otherwise, M and N have eager normal forms M' and N'; i.e., $M \Downarrow M'$ and $N \Downarrow N'$. Therefore by Lemma 24 and validity of β_{v}-reduction, $\mathcal{O}[\![M']\!] \approx \mathcal{O}[\![N']\!]$. Since M' is in eager normal form, by Proposition 27, either $\mathcal{O}[\![M']\!]\langle p\rangle \xrightarrow{\overline{x}(z,q)} P$ or $\mathcal{O}[\![M']\!]\langle p\rangle \xrightarrow{\overline{p}(y)} P$, and likewise for N'. This yields two cases:

   (a) $M' = C_{\mathrm{e}}[xV]$, $N' = C'_{\mathrm{e}}[xV']$, and

   $$\mathcal{O}_{\mathrm{V}}[\![V]\!]\langle z\rangle \mid q(y).\,\mathcal{O}[\![C_{\mathrm{e}}[y]]\!]\langle p\rangle \quad \approx \quad \mathcal{O}_{\mathrm{V}}[\![V']\!]\langle z\rangle \mid q(y).\,\mathcal{O}[\![C'_{\mathrm{e}}[y]]\!]\langle p\rangle\ .$$

   We observe that name q does not appear free in $\mathcal{O}_{\mathrm{V}}[\![V]\!]\langle z\rangle$ or $\mathcal{O}_{\mathrm{V}}[\![V']\!]\langle z\rangle$, hence by Lemma 30, $\mathcal{O}_{\mathrm{V}}[\![V]\!] \approx \mathcal{O}_{\mathrm{V}}[\![V']\!]$, thus $\mathcal{O}[\![V]\!] \approx \mathcal{O}[\![V']\!]$.
   Likewise, z' does not appear free in either $q(y).\,\mathcal{O}[\![C_{\mathrm{e}}[y]]\!]\langle p\rangle$ or $q(y).\,\mathcal{O}[\![C'_{\mathrm{e}}[y]]\!]\langle p\rangle$, and both $\mathcal{O}_{\mathrm{V}}[\![V]\!]\langle z\rangle$ and $\mathcal{O}_{\mathrm{V}}[\![V']\!]\langle z\rangle$ are prefixed by a replicated input on z. Hence, by Lemma 30, $\mathcal{O}[\![C_{\mathrm{e}}[y]]\!] \approx \mathcal{O}[\![C'_{\mathrm{e}}[y]]\!]$.
   Therefore $V\ \mathcal{R}\ V'$ and $C_{\mathrm{e}}[y]\ \mathcal{R}\ C'_{\mathrm{e}}[y]$.

   (b) M' and N' are values. They can be abstractions or variables.

      i. If both are abstractions $M' = \lambda z.\,M''$, $N' = \lambda z.\,N''$, and

      $$!y(z, q).\,\mathcal{O}[\![M'']\!]\langle q\rangle \approx\ !y(z, q).\,\mathcal{O}[\![N'']\!]\langle q\rangle,$$

      hence, by Lemma 31, $\mathcal{O}[\![M'']\!] \approx \mathcal{O}[\![N'']\!]$, which gives $M''\ \mathcal{R}\ N''$.

      ii. If both are variables, as seen above, we necessarily have $M' = N' = x$. We have $x\ \mathcal{R}\ x$, thus we can conclude.

      iii. Otherwise, assume $M' = \lambda z.\,M''$ and $N' = x$ without loss of generality. Then $\mathcal{O}[\![M']\!] \approx \mathcal{O}[\![N']\!] \approx \mathcal{O}[\![x]\!]$. By Lemma 32, $M'' \Downarrow C_{\mathrm{e}}[xV]$ for some C_{e}, V, and also $\mathcal{O}[\![V]\!] \approx \mathcal{O}[\![z]\!]$ and $\mathcal{O}[\![C_{\mathrm{e}}[y]]\!] \approx \mathcal{O}[\![y]\!]$ for some y fresh.
      Hence, $V\ \mathcal{R}\ z$, $C_{\mathrm{e}}[y]\ \mathcal{R}\ y$ for some y fresh, and we can conclude using case 6 of Definition 6.

□

4.4. Completeness and Full Abstraction

Suppose $M \Leftrightarrow_\eta N$. Then there is an η-eager normal-form bisimulation $\mathcal{R}$ such that $M \mathcal{R} N$. The completeness of the encoding can thus be stated as follows: given $\mathcal{R}$ an η-eager normal-form bisimulation, for all $(M, N) \in \mathcal{R}$, $\mathcal{I}[\![M]\!] \approx \mathcal{I}[\![N]\!]$.

To increase readability of the proof, we first show completeness for $\Leftrightarrow$, rather than $\Leftrightarrow_\eta$.

$$\begin{array}{ll}
M \Uparrow \text{ and } N \Uparrow: & X_{M,N} = (\widetilde{y})\ \mathcal{I}[\![\Omega]\!] \\
M \Downarrow C_{\mathrm{e}}[xV] \text{ and } N \Downarrow C'_{\mathrm{e}}[xV']: & X_{M,N} = (\widetilde{y})\ \mathcal{I}[\![(\lambda z.\, X_{C_{\mathrm{e}}[z],C'_{\mathrm{e}}[z]})\ (x\ X_{V,V'})]\!] \\
M \Downarrow x \text{ and } N \Downarrow x: & X_{M,N} = (\widetilde{y})\ \mathcal{I}[\![x]\!] \\
M \Downarrow \lambda x.\, M' \text{ and } N \Downarrow \lambda x.\, N': & X_{M,N} = (\widetilde{y})\ \mathcal{I}[\![\lambda x.\, X_{M',N'}]\!] \\
M \Downarrow x,\ N \Downarrow \lambda z.\, N',\ N' \Downarrow C_{\mathrm{e}}[xV]: & X_{M,N} = (\widetilde{y})\ \mathcal{I}[\![\lambda z.\, \big((\lambda w.\, X_{w,C_{\mathrm{e}}[w]})\ (x\ X_{z,V})\big)]\!] \\
M \Downarrow \lambda z.\, M',\ M' \Downarrow C_{\mathrm{e}}[xV],\ N \Downarrow x: & X_{M,N} = (\widetilde{y})\ \mathcal{I}[\![\lambda z.\, \big((\lambda w.\, X_{C_{\mathrm{e}}[w],w})\ (x\ X_{V,z})\big)]\!]
\end{array}$$

Figure 4: System $\mathcal{E}_{\mathcal{R}}$ of equations (the last two equations are included only when considering $\leftrightharpoons_\eta$)

Introducing systems of equations. Suppose $\mathcal{R}$ is an eager normal-form bisimulation. We define an (infinite) system of equations $\mathcal{E}_{\mathcal{R}}$, solutions of which will be obtained from the encodings of the pairs in $\mathcal{R}$. The definition of $\mathcal{E}_{\mathcal{R}}$ is given on Figure 4. We then use Theorem 16 and Lemma 18 to show that $\mathcal{E}_{\mathcal{R}}$ has a unique solution.

We assume an ordering on names and variables, so to be able to view (finite) sets of these as tuples. In the equations of Figure 4, $\widetilde{y}$ is assumed to be the ordering of $\mathsf{fv}(M, N)$. Moreover, if F is an abstraction, say $(\widetilde{a})\ P$, then $(\widetilde{y})\ F$ is an abbreviation for its uncurrying $(\widetilde{y}, \widetilde{a})\ P$.

There is one equation $X_{M,N} = E_{M,N}$ for each pair $(M, N) \in \mathcal{R}$. The body $E_{M,N}$ is essentially the encoding of the eager normal form (or absence thereof) of M and N, with the variables of the equations representing the coinductive hypothesis. To formalise this, we extend the encoding of the λ-calculus to equation variables by setting

$$\mathcal{I}[\![X_{M,N}]\!] \overset{\text{def}}{=} (p)\ X_{M,N}\langle \widetilde{y}, p\rangle \qquad \text{where } \widetilde{y} = \mathsf{fv}(M, N)\ .$$

Systems $\mathcal{E}_{\mathcal{R}}$ and $\mathcal{E}'_{\mathcal{R}}$. We introduce two systems of equations, $\mathcal{E}_{\mathcal{R}}$ (Figure 4) and $\mathcal{E}'_{\mathcal{R}}$ (Figure 5). On both figures, we provide the equations which are needed to handle $\leftrightharpoons_\eta$. The systems to handle $\leftrightharpoons$ are obtained by omitting some equations (precisely, the last two equations in Figures 4 and 5).

Given $(M, N) \in \mathcal{R}$, we now comment on the equation $X_{M,N} = E_{M,N}$ in system $\mathcal{E}_{\mathcal{R}}$. The equation is parametrised on the free variables of M and N (to ensure that the body $E_{M,N}$ is a name-closed abstraction) and an additional continuation name (as all encodings of terms). Accordingly, we write $\widetilde{y}$ for the ordering of $\mathsf{fv}(M, N)$.

1. If $M \Uparrow$ and $N \Uparrow$, then the equation is

$$X_{M,N} = (\widetilde{y})\ \mathcal{I}[\![\Omega]\!]$$

   (We could use $(\widetilde{y}, p)\ \mathbf{0}$ above, since the encoding of a diverging term is bisimilar to $\mathbf{0}$).

2. If $M \Downarrow x$ and $N \Downarrow x$, then the equation is the encoding of x:

$$X_{M,N} = (\widetilde{y})\ \mathcal{I}[\![x]\!] = (\widetilde{y}, p)\ \overline{p}(z).\, z \triangleright x$$

   Since x is the eager normal form of M and N, $x \in \widetilde{y}$. Note that $\widetilde{y}$ can contain more names, occurring free in M or N.

3. If $M \Downarrow \lambda x.\, M'$ and $N \Downarrow \lambda x.\, N'$, then the equation encodes an abstraction whose body refers to the normal forms of M', N', via the variable $X_{M',N'}$:

$$\begin{array}{rcl}
X_{M,N} & = & (\widetilde{y})\ \mathcal{I}[\![\lambda x.\, X_{M',N'}]\!] \\
& = & (\widetilde{y}, p)\ \overline{p}(z).\, !z(x, q).\, X_{M',N'}\langle \widetilde{y}', q\rangle,
\end{array}$$

   where $\widetilde{y}'$ is the ordering of $\mathsf{fv}(M', N')$.

4. If $M \Downarrow C_{\mathrm{e}}[xV]$ and $N \Downarrow C'_{\mathrm{e}}[xV']$, we separate the evaluation contexts and the values, as in Definition 3. In the body of the equation, this is achieved by: (i) rewriting $C_{\mathrm{e}}[xV]$ into $(\lambda z.\, C_{\mathrm{e}}[z])(xV)$, for some fresh z, and similarly for C'_{e} and V' (such a transformation is valid for $\leftrightharpoons$); and (ii) referring to the variable for the evaluation contexts, $X_{C_{\mathrm{e}}[z],C'_{\mathrm{e}}[z]}$, and to the variable for the values, $X_{V,V'}$. This yields the equation (for z fresh):

$$X_{M,N} = (\widetilde{y})\ \mathcal{I}[\![(\lambda z.\, X_{C_{\mathrm{e}}[z],C'_{\mathrm{e}}[z]})\ (x\ X_{V,V'})]\!]$$

As an example, suppose $(I, \lambda x.\, M) \in \mathcal{R}$, where $I = \lambda x.\, x$ and $M = (\lambda zy.\, z)xx'$. We obtain the following equations: (we have $\mathsf{fv}(M) = \{x, x'\}$, and we assume x is before x' in the ordering of variables):

1. $$\begin{aligned} X_{I,\lambda x.M} &= (x')\ \mathcal{I}[\![\lambda x.\, X_{x,M}]\!] \\ &= (x', p)\ \overline{p}(y).\, !y(x,q).\, X_{x,M}\langle x, x', q\rangle \end{aligned}$$

2. $$\begin{aligned} X_{x,M} &= (x, x')\ \mathcal{I}[\![x]\!] \\ &= (x, x', p)\ \overline{p}(y).\, y \triangleright x \end{aligned}$$

Before explaining how $\mathcal{R}$ yields solutions of the system, we prove the following important law:

Lemma 34. *If C_{e} is an evaluation context, V is a value, x is a name and z is fresh in C_{e}, then*

$$\mathcal{I}[\![C_{\mathrm{e}}[xV]]\!] \approx \mathcal{I}[\![(\lambda z.\, C_{\mathrm{e}}[z])(xV)]\!].$$

Proof. By induction on the evaluation context C_{e}. When $C_{\mathrm{e}} = [\cdot]$, we show that $\mathcal{I}[\![xV]\!] \approx \mathcal{I}[\![I(xV)]\!]$ by algebraic reasoning. The other cases can be handled similarly. □

Solutions of $\mathcal{E}_{\mathcal{R}}$. Having set the system of equations for $\mathcal{R}$, we now define solutions for it from the encoding of the pairs in $\mathcal{R}$.

We can view the relation $\mathcal{R}$ as an ordered sequence of pairs (e.g., assuming some lexicographical ordering). Then $\mathcal{R}_i$ indicates the tuple obtained by projecting the pairs in $\mathcal{R}$ onto the i-th component ($i = 1, 2$). Moreover (M_j, N_j) is the j-th pair in $\mathcal{R}$, and $\widetilde{y_j}$ is $\mathsf{fv}(M_j, N_j)$.

We write $\mathcal{I}^{\mathrm{c}}[\![\mathcal{R}_1]\!]$ for the closed abstractions resulting from the encoding of $\mathcal{R}_1$, i.e., the tuple whose j-th component is $(\widetilde{y_j})\ \mathcal{I}[\![M_j]\!]$, and similarly for $\mathcal{I}^{\mathrm{c}}[\![\mathcal{R}_2]\!]$.

Definition 35. We define $\mathcal{I}^{\mathrm{c}}[\![\mathcal{R}_1]\!] := \{(\widetilde{y})\ \mathcal{I}[\![M]\!] \mid \exists N, M\ \mathcal{R}\ N \text{ and } \widetilde{y} = \mathsf{fv}(M,N)\}$. $\mathcal{I}^{\mathrm{c}}[\![\mathcal{R}_2]\!]$ is defined similarly, based on the right-hand side of the relation.

We observe that if $\widetilde{y} = \mathsf{fv}(M, N)$, then $(\widetilde{y}, p)\ \mathcal{I}[\![M]\!]\langle p\rangle$ and $(\widetilde{y}, p)\ \mathcal{I}[\![N]\!]\langle p\rangle$ are closed abstractions.

As a direct consequence of Lemmas 24 and 25, we obtain the following result, which is used below.

Lemma 36 (Validity of β_{v}-reduction for the original encoding)**.** *For any M, N in Λ, $M \longrightarrow N$ implies that for any p, $\mathcal{I}[\![M]\!]\langle p\rangle \approx \mathcal{I}[\![N]\!]\langle p\rangle$.*

Lemma 37. *$\mathcal{I}^{\mathrm{c}}[\![\mathcal{R}_1]\!]$ and $\mathcal{I}^{\mathrm{c}}[\![\mathcal{R}_2]\!]$ are solutions of the system of equations $\mathcal{E}_{\mathcal{R}}$.*

Proof. We only show the property for $\mathcal{R}_1$, the case for $\mathcal{R}_2$ is handled similarly.

We show that each component of $\mathcal{I}^{\mathrm{c}}[\![\mathcal{R}_1]\!]$ is solution of the corresponding equation, i.e., for the j-th component we show $(\widetilde{y_j})\ \mathcal{I}[\![M_j]\!] \approx E_{M_j,N_j}[\mathcal{I}^{\mathrm{c}}[\![\mathcal{R}_1]\!]]$.

We reason by cases over the shape of the eager normal form of M_j, N_j.

- If $M \Uparrow$, we use Lemma 29, which gives us $(\widetilde{y})\ \mathcal{I}[\![M]\!] \approx (\widetilde{y}, p)\ \mathbf{0} \approx (\widetilde{y})\ \mathcal{I}[\![\Omega]\!]$.
- If $M \Downarrow C_{\mathrm{e}}[xV]$, we have to show that:

  $$\mathcal{I}[\![M]\!] \approx \mathcal{I}[\![(\lambda z.\, C_{\mathrm{e}}[z])(xV)]\!]\ .$$

  By Lemma 36, $\mathcal{I}[\![M]\!] \approx \mathcal{I}[\![C_{\mathrm{e}}[xV]]\!]$; we then conclude by Lemma 34.
- If $M \Downarrow \lambda x.\, M'$ (and N also reduces to an abstraction), then:

  $$\begin{aligned} E_{M,N}[\mathcal{I}[\![\mathcal{R}_1]\!]]\langle \widetilde{y}, p\rangle &= \overline{p}(z).\, !z(x,q).\, \mathcal{I}[\![M']\!]\langle q\rangle \\ &\approx \mathcal{I}[\![M]\!]\langle \widetilde{y}, p\rangle \text{ (by Lemma 36)}\ . \end{aligned}$$

- If $M \Downarrow x$ (and $N \Downarrow x$), again:

  $$\begin{aligned} E_{M,N}[\mathcal{I}[\![\mathcal{R}_1]\!]]\langle \widetilde{y}, p\rangle &= \mathcal{I}[\![x]\!]\langle p\rangle \\ &\approx \mathcal{I}[\![M]\!]\langle \widetilde{y}, p\rangle \text{ (by Lemma 36)}\ . \end{aligned}$$

□

$$
\begin{array}{lr}
M \Uparrow \text{ and } N \Uparrow: & X_{M,N} = (\widetilde{y}, p)\ \mathbf{0} \\
M \Downarrow C_{\mathrm{e}}[xv] \text{ and } N \Downarrow C'_{\mathrm{e}}[xv'] : & X_{M,N} = (\widetilde{y}, p)\ \overline{x}(z, q).\\
& (X^{\mathcal{V}}_{V,V'}\langle z, \widetilde{y}'\rangle \mid q(w).\, X_{C_{\mathrm{e}}[w],C'_{\mathrm{e}}[w]}\langle \widetilde{y}'', p\rangle) \\
M \Downarrow V \text{ and } N \Downarrow V' : & X_{M,N} = (\widetilde{y}, p)\ \overline{p}(y).\, X^{\mathcal{V}}_{v,v'}\langle z, \widetilde{y}'\rangle \\
V = x \text{ and } V' = x : & X^{\mathcal{V}}_{x,x} = (z, x)\ z \triangleright x \\
V = \lambda x.\, M \text{ and } V' = \lambda x.\, N : & X^{\mathcal{V}}_{\lambda x.M,\lambda x.N} = (z, \widetilde{y})\ !z(x, q).\, X_{M,N}\langle \widetilde{y}', q\rangle \\
V = x,\ V' = \lambda z.\, N,\ N \Downarrow C_{\mathrm{e}}[xV] : & X^{\mathcal{V}}_{x,\lambda z.N} = (y_0, \widetilde{y})\ !y_0(z, q).\, \overline{x}(z', q').\\
& (X^{\mathcal{V}}_{z,V}\langle z', \widetilde{y}'\rangle \mid q'(w).\, X_{w,C_{\mathrm{e}}[w]}\langle \widetilde{y}'', q\rangle) \\
V = \lambda z.\, M,\ M \Downarrow C_{\mathrm{e}}[xV],\ V' = x : & X^{\mathcal{V}}_{\lambda z.M,x} = (y_0, \widetilde{y})\ !y_0(z, q).\, \overline{x}(z', q').\\
& (X^{\mathcal{V}}_{V,z}\langle z', \widetilde{y}'\rangle \mid q'(w).\, X_{C_{\mathrm{e}}[w],w}\langle \widetilde{y}'', q\rangle)
\end{array}
$$

Figure 5: System $\mathcal{E}'_{\mathcal{R}}$ of equations (the last two equations are included only when considering $\leftrightharpoons_\eta$)

Unique solution for $\mathcal{E}_{\mathcal{R}}$. We rely on Theorem 16 to prove uniqueness of solutions for $\mathcal{E}_{\mathcal{R}}$. The only delicate requirement is the one on divergence for the syntactic solution. For this, we use Lemma 18. We thus introduce an auxiliary system of equations, $\mathcal{E}'_{\mathcal{R}}$, that extends $\mathcal{E}_{\mathcal{R}}$, and whose syntactic solutions have no τ-transition and hence trivially satisfy the requirement. The definition of $\mathcal{E}'_{\mathcal{R}}$ is presented in Figure 5. Like the original system $\mathcal{E}_{\mathcal{R}}$, so the new one $\mathcal{E}'_{\mathcal{R}}$ is defined by inspection of the pairs in $\mathcal{R}$; in $\mathcal{E}'_{\mathcal{R}}$, however, a pair of $\mathcal{R}$ may sometimes yield more than one equation. Thus, let $(M, N) \in \mathcal{R}$ with $\widetilde{y} = \mathsf{fv}(M, N)$ (we also write $\widetilde{y}'$ or $\widetilde{y}''$ for the free variables of the terms indexing the corresponding equation variable).

1. When $M \Uparrow$ and $N \Uparrow$, the equation is
$$X_{M,N} = (\widetilde{y}, p)\ \mathbf{0}\ .$$

2. When $M \Downarrow V$ and $N \Downarrow V'$, we introduce a new equation variable $X^{\mathcal{V}}_{V,V'}$ and a new equation; this will allow us, in the following step (3), to perform some optimisations. The equation is
$$X_{M,N} = (\widetilde{y}, p)\ \overline{p}(z).\, X^{\mathcal{V}}_{V,V'}\langle z, \widetilde{y}'\rangle\ ,$$
and we have, accordingly, the two following additional equations corresponding to the cases where values are functions or variables:
$$
\begin{array}{rcl}
X^{\mathcal{V}}_{\lambda x.M',\lambda x.N'} & = & (z, \widetilde{y})\ !z(x, q).\, X_{M',N'}\langle \widetilde{y}', q\rangle \\
X^{\mathcal{V}}_{x,x} & = & (z, x)\ z \triangleright x
\end{array}
$$

3. When $M \Downarrow C_{\mathrm{e}}[xV]$ and $N \Downarrow C_{\mathrm{e}}[xV']$, we refer to $X^{\mathcal{V}}_{V,V'}$, instead of $X_{V,V'}$, so to remove all initial reductions in the corresponding equation for $\mathcal{E}_{\mathcal{R}}$. The first action thus becomes an output:
$$X_{M,N} = (\widetilde{y}, p)\ \overline{x}(z, q).\, (X^{\mathcal{V}}_{V,V'}\langle z, \widetilde{y}'\rangle \mid q(w).\, X_{C_{\mathrm{e}}[w],C'_{\mathrm{e}}[w]}\langle \widetilde{y}'', p\rangle)$$

Lemmas 38 and 39 are needed to apply Lemma 18. (In Lemma 38, 'extend' is as by Definition 17.)

Lemma 38. *The system of equations $\mathcal{E}'_{\mathcal{R}}$ extends the system of equations $\mathcal{E}_{\mathcal{R}}$.*

Proof. The new system $\mathcal{E}'_{\mathcal{R}}$ is obtained from $\mathcal{E}_{\mathcal{R}}$ by modifying the equations and adding new ones. More precisely, whenever M and N are values, an additional equation is introduced, using a variable written $X^{\mathcal{V}}$. A solution for the extended system yields a solution for the original system by looking only at equation variables which are not of the form $X^{\mathcal{V}}_{V,V'}$. □

Lemma 39. *$\mathcal{E}'_{\mathcal{R}}$ has a unique solution.*

Proof. Divergence-freedom for the syntactic solutions of $\mathcal{E}'_{\mathcal{R}}$ holds because in the equations each name (bound or free) can appear either only in inputs or only in outputs. Indeed, in the syntactic solutions of $\mathcal{E}'_{\mathcal{R}}$, linear names $(p, q, \dots)$ are used exactly once in subject position, and non-linear names $(x, y, w, \dots)$, when used in subject position, are either used exclusively in input or exclusively in output.

As a consequence, since the labelled transition system is ground, no τ-transition can ever be performed, after any number of visible actions. Further, $\mathcal{E}'_{\mathcal{R}}$ is guarded. Hence we can apply Theorem 16.

$\square$

Hence, by Lemma 18, $\mathcal{E}_{\mathcal{R}}$ has a unique solution.

We can observe that $\mathcal{E}_{\mathcal{R}}$ has equations of the form $X = \mathcal{I}[\![\Omega]\!]$ associated to a diverging λ-term. Such equations give rise to *innocuous divergences*, using the terminology of [7]. A refined version of Theorem 16 is stated in [7], in order to handle such divergences; this would make it possible to avoid using Lemma 18, at the cost of a more intricate setting. Using Lemma 18 allows us to keep the framework simpler.

A more direct proof of Lemma 39 would have been possible, by reasoning coinductively over the η-eager normal-form bisimulation defining the system of equations.

Lemma 40 (Completeness for $\Leftrightarrow$)**.** *$M \Leftrightarrow N$ implies $\mathcal{I}[\![M]\!] \approx \mathcal{I}[\![N]\!]$, for any $M, N \in \Lambda$.*

Proof. Consider an eager normal-form bisimulation $\mathcal{R}$, and the corresponding systems of equations $\mathcal{E}_{\mathcal{R}}$ and $\mathcal{E}'_{\mathcal{R}}$. Lemmas 39 and 38 allow us to apply Lemma 18 and deduce that $\mathcal{E}_{\mathcal{R}}$ has a unique solution. By Lemma 37, $\mathcal{I}^{c}[\![\mathcal{R}_1]\!]$ and $\mathcal{I}^{c}[\![\mathcal{R}_2]\!]$ are solutions of $\mathcal{E}_{\mathcal{R}}$. Thus, from $M \;\mathcal{R}\; N$, we deduce $(\widetilde{y})\; \mathcal{I}[\![M]\!] \approx (\widetilde{y})\; \mathcal{I}[\![N]\!]$, where $\widetilde{y} = \mathsf{fv}(M, N)$. Hence also $\mathcal{I}[\![M]\!] \approx \mathcal{I}[\![N]\!]$.

$\square$

Completeness for $\Leftrightarrow_\eta$. The proof for $\Leftrightarrow$ is extended to $\Leftrightarrow_\eta$, maintaining its structure. We highlight the main additional reasoning steps.

We enrich $\mathcal{E}_{\mathcal{R}}$ with the equations corresponding to the two additional clauses of $\Leftrightarrow_\eta$ (Definition 6). When $M \Downarrow x$ and $N \Downarrow \lambda z.\, N'$, where $N' \Leftrightarrow_\eta xz$, we proceed as in case 4 of the definition of $\mathcal{E}_{\mathcal{R}}$, given that $N \Leftrightarrow_\eta \lambda z.\, ((\lambda w.\, C_{\mathrm{e}}[w])(xV))$. The equation is:

$$X_{M,N} = (\widetilde{y})\; \mathcal{I}[\![\lambda z.\, \big((\lambda w.\, X_{w,C_{\mathrm{e}}[w]})\; (x\; X_{z,V})\big)]\!] \;\; .$$

We proceed likewise in the symmetric case.

In the optimised equations, we add the following equation (relating values), as well as its symmetric counterpart:

$$X^{\mathcal{V}}_{x,\lambda z.N'} = (y_0, \widetilde{y})\; !y_0(z,q).\, \overline{x}(z',q').\, (X^{\mathcal{V}}_{z,V}\langle z', \widetilde{y}'\rangle \mid q'(w).\, X_{w,C_{\mathrm{e}}[w]}\langle \widetilde{y}'', q\rangle) \;\; .$$

We follow the approach of Lemmas 38 and 39 to show unique solution for $\mathcal{E}_{\mathcal{R}}$.

The following lemma validates η-expansion for the encoding, and is useful below. It is proved using algebraic reasoning and the definition of links.

Lemma 41. *$\mathcal{I}[\![\lambda y.\, xy]\!] \approx \mathcal{I}[\![x]\!]$.*

Finally, we prove that $\mathcal{I}^{c}[\![\mathcal{R}_1]\!]$ and $\mathcal{I}^{c}[\![\mathcal{R}_2]\!]$ are solutions of $\mathcal{E}_{\mathcal{R}}$. Two additional cases are to be considered:

- If $M \Downarrow x$ and $N \Downarrow \lambda y.\, N'$, then:

$$\begin{aligned} E_{M,N}[\mathcal{I}[\![\mathcal{R}_1]\!]]\langle \widetilde{y}, p\rangle &= \overline{p}(z).\, !z(y,q).\, \mathcal{I}[\![(\lambda w.\, w)(xy)]\!]\langle q\rangle \\ &\approx \overline{p}(z).\, !z(y,q).\, \mathcal{I}[\![xy]\!]\langle q\rangle \text{ (by Lemma 34)} \\ &= \mathcal{I}[\![\lambda y.\, xy]\!]\langle q\rangle \\ &\approx \mathcal{I}[\![x]\!]\langle p\rangle \text{ (by Lemma 41)} \\ &\approx \mathcal{I}[\![M]\!]\langle \widetilde{y}, p\rangle \text{ (by Lemma 36)} \;\; . \end{aligned}$$

- If $M \Downarrow \lambda y.\, M'$, $M' \Downarrow C_{\mathrm{e}}[xV]$ and $N \Downarrow x$, then:

$$\begin{aligned} E_{M,N}[\mathcal{I}[\![\mathcal{R}_1]\!]]\langle \widetilde{y}, p\rangle &= \overline{p}(z).\, !z(y,q).\, \mathcal{I}[\![(\lambda w.\, C_{\mathrm{e}}[w])(xV)]\!]\langle q\rangle \\ &\approx \overline{p}(z).\, !z(y,q).\, \mathcal{I}[\![C_{\mathrm{e}}[xV]]\!]\langle q\rangle \text{ (by Lemma 34)} \\ &= \mathcal{I}[\![\lambda y.\, C_{\mathrm{e}}[xV]]\!]\langle q\rangle \\ &\approx \mathcal{I}[\![M]\!]\langle \widetilde{y}, p\rangle \text{ (by Lemma 36)} \ . \end{aligned}$$

Given the previous results, we can reason as for the proof of Lemma 40 to establish completeness.

Proposition 42 (Completeness for $\Leftrightarrow_\eta$)**.** *For any M, N in Λ, $M \Leftrightarrow_\eta N$ implies $\mathcal{I}[\![M]\!] \approx \mathcal{I}[\![N]\!]$.*

Combining Propositions 33 and 42, and Theorem 12, we deduce full abstraction for $\Leftrightarrow_\eta$ with respect to barbed congruence.

Theorem 43 (Full Abstraction for $\Leftrightarrow_\eta$)**.** *For any M, N in Λ, we have $M \Leftrightarrow_\eta N$ iff $\mathcal{I}[\![M]\!] \simeq^{\mathrm{I}\pi} \mathcal{I}[\![N]\!]$*

Remark 44 (Unique solutions versus up-to techniques)**.** For Milner's encoding of call-by-name λ-calculus, the completeness part of the full abstraction result with respect to Levy-Longo Trees [28] relies on *up-to techniques for bisimilarity*. Precisely, given a relation $\mathcal{R}$ on λ-terms that represents a tree bisimulation, one shows that the π-calculus encoding of $\mathcal{R}$ is a π-calculus bisimulation *up-to context and expansion*.

In the up-to technique, expansion is used to manipulate the derivatives of two transitions so to bring up a common context. Such up-to technique is not powerful enough for the call-by-value encoding and the Eager Trees because some of the required transformations would violate expansion (i.e., they would require to replace a term by a 'less efficient' one). An example of this is the law proved in Lemma 37, that would have to be applied from right to left so to implement the branching in clause (2) of Definition 3 (as a context with two holes).

The use of the technique of unique solution of equations allows us to overcome the problem: the law in Lemma 37 and similar laws that introduce 'inefficiencies' can be used (and they are indeed used, in various places), as long as they do not produce new divergences.

5. Encoding into ALπ

Full abstraction with respect to η-Eager-Tree equality also holds for Milner's simplest encoding, namely $\mathcal{V}$ (Section 3), provided that the target language of the encoding is taken to be ALπ (see Section 1.5). The adoption of ALπ implicitly allows us to control capabilities, avoiding violations of laws such as (4) in the Introduction. In ALπ, bound output prefixes such as $\overline{a}(x).\, x(y)$ are abbreviations for $\boldsymbol{\nu} x\, (\overline{a}\langle x\rangle \mid x(y))$.

5.1. The Local π-calculus

We present here the results which make it possible for us to apply the unique-solution technique to ALπ. The main idea is to exploit a characterisation of barbed congruence as ground bisimilarity. However, to obtain this, ground bisimilarity has to be set on top of a non-standard transition system, specialised to ALπ [17]. The Labelled Transition System (LTS) is produced by the rules in Figure 6; these modify ordinary transitions (the $\xrightarrow{\mu}$ relation) by adding *static links* $a \blacktriangleright b$, which are abbreviations defined thus:

$$a \blacktriangleright b \stackrel{\text{def}}{=} !a(\widetilde{x}).\, \overline{b}\langle \widetilde{x}\rangle \ .$$

(We call them *static* links, following the terminology in [17], so to distinguish them from the links $a \triangleright b$ used in Iπ, whose definition makes use of recursive process definitions — static links only need replication.)

Notations for the ordinary LTS ($\xrightarrow{\mu}$) are transported onto the new LTS ($\xmapsto{\mu}$), yielding, e.g., transitions $\stackrel{\mu}{\Longmapsto}$ and $\stackrel{\hat{\mu}}{\Longmapsto}$.

We write $\overset{\rightarrow}{\approx}$ for (ground) bisimilarity on the new LTS, defined as $\approx$ in Definition 10, but using the new LTS in place of the ordinary one. Barbed congruence in ALπ, $\simeq^{\mathrm{AL}\pi}$, is defined as

$$\frac{P \xrightarrow{\boldsymbol{\nu}\widetilde{d}\,\overline{a}\langle\widetilde{b}\rangle} P' \qquad \widetilde{c}\cap(\mathrm{fn}(P)\cup\widetilde{d})=\emptyset}{P \xmapsto{\boldsymbol{\nu}\widetilde{c}\,\overline{a}\langle\widetilde{c}\rangle} \boldsymbol{\nu}\widetilde{d}\,(\widetilde{c} \blacktriangleright \widetilde{b} \mid P')} \qquad \frac{P \xrightarrow{a(\widetilde{b})} P'}{P \xmapsto{a(\widetilde{b})} P'} \qquad \frac{P \xrightarrow{\tau} P'}{P \xmapsto{\tau} P'}$$

Figure 6: The modified labelled transition system for ALπ

by Definition 8 (on τ-transitions, which are the only transitions needed to define $\simeq^{\mathrm{AL}\pi}$, the new LTS and the original one coincide).

We present the definition of asynchronous (ground) bisimilarity, which is used in [17] to derive a characterisation of barbed congruence; asynchrony is needed because the calculus is asynchronous, and barbed congruence observes only output actions.

Definition 45 (Asynchronous bisimilarity)**.** Asynchronous bisimilarity, written $\overset{\mapsto}{\approx}_{\mathrm{a}}$ is the largest symmetric relation $\mathcal{R}$ such that $P\mathcal{R}Q$ implies

- if $P \overset{\mu}{\mapsto} P'$ and μ is not an input, then there is Q' s.t. $Q \overset{\hat{\mu}}{\Longmapsto} Q'$ and $P'\mathcal{R}Q'$, and
- if $P \xmapsto{a(\widetilde{b})} P'$, then either $Q \overset{a(\widetilde{b})}{\Longmapsto} Q'$ and $P'\mathcal{R}Q'$ for some Q', or $Q \Rightarrow Q'$ and $P'\mathcal{R}(Q' \mid \overline{a}\langle\widetilde{b}\rangle)$ for some Q'.

Theorem 46 ([17])**.** *On ALπ processes that are image-finite up to $\approx$, relations $\overset{\mapsto}{\approx}_{\mathrm{a}}$ and $\simeq^{\mathrm{AL}\pi}$ coincide.*

To apply our technique of unique solutions of equations it is however convenient to use *synchronous* bisimilarity. The following result allows us to do so:

Theorem 47. *On ALπ processes that are image-finite up to $\approx$ and have no input on a free name, relations $\overset{\mapsto}{\approx}$ and $\overset{\mapsto}{\approx}_{\mathrm{a}}$ coincide.*

Proof. By construction, $\overset{\mapsto}{\approx} \subseteq \overset{\mapsto}{\approx}_{\mathrm{a}}$

To show that $\overset{\mapsto}{\approx}_{\mathrm{a}} \subseteq \overset{\mapsto}{\approx}$, we first establish a property about output capability and transitions. We say that P *respects output capability* if any free name used in input subject position in P may not be used in output, either in subject or object position, in P. We show that if P respects output capability and $P \overset{\mu}{\mapsto} P'$, then so does P'.

We reason on the type of the transition $P \xrightarrow{\mu'} P'$ from which $P \overset{\mu}{\mapsto} P'$ is derived. For simplicity, we consider only monadic actions.

1. if $P \xrightarrow{a(c)} P'$: then c is fresh for P, and cannot be used in input in P'. The property hence holds.
2. if $P \xrightarrow{\tau} P'$: by hypothesis, the communication takes place at a restricted name (otherwise the name would have free occurrences in input and in output in P). If the transmitted name is restricted as well, there is nothing to prove. Otherwise, let us suppose, to illustrate the reasoning, that $P = \boldsymbol{\nu}a\,(a(b).\,P_0 \mid \overline{a}c) \xrightarrow{\tau} \boldsymbol{\nu}a\;P_0\{c/b\}$ (the transmitted name is c). Because c occurs free in output in P, it cannot occur in input, since P respects output capability. Because we are in ALπ, the new occurrences of c created by the substitution $\{c/b\}$ are in output position. So c cannot occur in input position in $P' = \boldsymbol{\nu}a\,P_0\{c/b\}$. The proof in the general case follows the same ideas.
3. if $P \xrightarrow{\overline{a}b} P'$, and $P \xmapsto{\overline{a}(c)} c \blacktriangleright b \mid P'$: because P respects output capability, so does P', as the transition $P \xrightarrow{\overline{a}b} P'$ does not introduce any new occurrence of names. Since b is used in output in P, it cannot be used in input in P, and hence P' does not contain output occurrences of b. We deduce that $c \blacktriangleright b \mid P'$ also respects output capability: indeed, c is fresh and thus is used only in input, and b is used only in output in $c \blacktriangleright b$.
4. if $P \xrightarrow{\overline{a}(b)} P'$, and $P \xmapsto{\overline{a}(c)} \boldsymbol{\nu}b\,(c \blacktriangleright b \mid P')$. This case is simpler than the previous one, since b is bound in the resulting process, and, as before, c is fresh and is only used in input.

Thus, we can consider bisimulations containing only processes that respect output capability.

Now, assume $\mathcal{R}$ is an asynchronous bisimulation relation containing only processes that respect output capability. We show that $\mathcal{R}$ is also a synchronous bisimulation relation.

The only interesting case is when $(P, Q) \in \mathcal{R}$, $P \overset{a(\widetilde{b})}{\longmapsto} P'$ and $Q \Longmapsto Q'$ for some Q' such that $P' \mathrel{\mathcal{R}} (Q' \mid \overline{a}\langle\widetilde{b}\rangle)$. We observe that $Q' \mid \overline{a}\langle\widetilde{b}\rangle$ can perform an output on a, which implies that P' has a free occurrence of a in output. Since by hypothesis $a \notin \widetilde{b}$, this means that b occurs free in input and in output in P, a contradiction.

Over processes that respect output capability, asynchronous bisimulation relations are synchronous bisimulation relations, thus $\overset{\rightarrow}{\approx}_{\mathrm{a}}$ and $\overset{\rightarrow}{\approx}$ coincide. This allows us to conclude, because ALπ processes that have no free input do respect output capability. $\square$

The property in Theorem 47 is new – we are not aware of papers in the literature presenting it. It is a consequence of the fact that, under the hypothesis of the theorem, and with a ground transition system, the only input actions in processes that can ever be produced are those emanating from the links, and two tested processes, if bisimilar, must have the same sets of (visible) links. We can moreover remark that in Theorem 47, the condition on inputs can be removed by adopting an asynchronous variant of bisimilarity; however, the synchronous version is easier to use in our proofs based on unique solution of equations.

For any $M \in \Lambda$ and p, process $\mathcal{V}[\![M]\!]\langle p\rangle$ is indeed image-finite up to $\approx$ and has no free input. From Theorems 46 and 47, we therefore deduce that $\overset{\rightarrow}{\approx}$ and $\simeq^{\mathrm{AL}\pi}$ coincide for processes obtained by the encoding $\mathcal{V}$.

5.2. *Full abstraction in ALπ*

We now discuss full abstraction for Milner's encoding $\mathcal{V}$, when the target language is ALπ. The proof of is overall very similar to that of Theorem 43.

The systems of equations for ALπ are similar to the ones we introduced for Iπ (Figures 2 and 3). They are presented in Appendix D. The equations defining the first system are exactly the same as in Figure 2, only encoding $\mathcal{V}$ is used instead of $\mathcal{I}$. The second system is an optimised version of the first. Again, it is defined as in Iπ; the differences are that we use static links.

The modified LTS of Figure 6 introduces additional static links with respect to the ground LTS. When establishing the counterpart of Lemmas 40 and 42, we need to reason about divergences, and must therefore show that these links do not produce new reductions.

Lemma 48. *Let P be an ALπ process such that P has no divergences in the ground LTS. Then it has no divergences in the modified LTS for ALπ.*

Proof. The replicated inputs guarding static links created by an output transition in the modified LTS are always at fresh names—the $\widetilde{c}$ in Figure 6. Hence, no communication at the names in $\widetilde{c}$ is possible. Furthermore, in the ground LTS, the additional inputs at the names in $\widetilde{c}$ are with fresh names, so they cannot generate new τ transitions. $\square$

The encoding into ALπ is fully abstract.

Theorem 49. *$M \leftrightarrows_\eta N$ iff $\mathcal{V}[\![M]\!] \simeq^{\mathrm{AL}\pi} \mathcal{V}[\![N]\!]$, for any $M, N \in \Lambda$.*

6. Contextual equivalence and preorders

We have presented full abstraction for η-Eager-Tree equality taking a 'branching' behavioural equivalence, namely barbed congruence, on the π-processes. We show here the same result for contextual equivalence, the most common 'linear' behavioural equivalence. We also extend the results to preorders.

We only discuss the encoding $\mathcal{I}$ into Iπ. Similar results however hold for the encoding $\mathcal{V}$ into ALπ.

6.1. Contextual relations and traces

Contextual equivalence is defined in the π-calculus analogously to its definition in the λ-calculus (Definition 2); thus, with respect to barbed congruence, the bisimulation game on reduction is dropped. Since we wish to handle preorders, we also introduce the *contextual preorder*.

Definition 50. Two Iπ processes P, Q are in the *contextual preorder*, written $P \lesssim_{\mathrm{ct}}^{\mathrm{I}\pi} Q$, if $C[P] \Downarrow_a$ implies $C[Q] \Downarrow_a$, for all contexts C. They are *contextually equivalent*, written $P \simeq_{\mathrm{ct}}^{\mathrm{I}\pi} Q$, if both $P \lesssim_{\mathrm{ct}}^{\mathrm{I}\pi} Q$ and $Q \lesssim_{\mathrm{ct}}^{\mathrm{I}\pi} P$ hold.

As usual, these relations are extended to abstractions by requiring instantiation of the parameters with fresh names. To manage contextual preorder and equivalence in proofs, we exploit characterisations of them as trace inclusion and equivalence. We define the traces of a process as follows:

Definition 51. A (finite, weak, ground) trace is a finite sequence of visible actions $\mu_1, \ldots, \mu_n$ such that for $i \neq j$ the bound names of μ_i and μ_j are all distinct and if $j < i$, the free names of μ_j and the bound names of μ_i are all distinct.

For $s = \mu_1, \ldots, \mu_n$, we write $P \stackrel{s}{\Longrightarrow}$ if $P \stackrel{\mu_1}{\Longrightarrow} P_1 \stackrel{\mu_2}{\Longrightarrow} P_2 \ldots P_{n-1} \stackrel{\mu_n}{\Longrightarrow} P_n$, for some processes $P_1, \ldots, P_n$. In such a situation, we sometimes say that s is a *finite trace* of P.

Definition 52. Two Iπ processes P, Q are in the *trace inclusion*, written $P \preceq_{\mathrm{tr}} Q$, if $P \stackrel{s}{\Longrightarrow}$ implies $Q \stackrel{s}{\Longrightarrow}$, for each trace s. They are *trace equivalent*, written $P \approx_{\mathrm{tr}} Q$, if both $P \preceq_{\mathrm{tr}} Q$ and $Q \preceq_{\mathrm{tr}} P$ hold.

The following result is standard.

Proposition 53. *In Iπ, relation $\lesssim_{\mathrm{ct}}^{\mathrm{I}\pi}$ coincides with $\preceq_{\mathrm{tr}}$, and relation $\simeq_{\mathrm{ct}}^{\mathrm{I}\pi}$ coincides with $\approx_{\mathrm{tr}}$.*

6.2. A proof technique for preorders

We modify the technique of unique solution of equations to reason about preorders, precisely the trace inclusion preorder.

We say that $\widetilde{F}$ is a *pre-fixed point for* $\preceq_{\mathrm{tr}}$ of a system of equations $\{\widetilde{X} = \widetilde{E}\}$ if $\widetilde{E}[\widetilde{F}] \preceq_{\mathrm{tr}} \widetilde{F}$; similarly, $\widetilde{F}$ is a *post-fixed point for* $\preceq_{\mathrm{tr}}$ if $\widetilde{F} \preceq_{\mathrm{tr}} \widetilde{E}[\widetilde{F}]$. In the case of equivalence, the technique of unique solutions exploits symmetry arguments, but symmetry does not hold for preorders. We overcome the problem by referring to the syntactic solution of the system in an asymmetric manner. This yields the two lemmas below, intuitively stating that the syntactic solution of a system is its smallest pre-fixed point, as well as, under the divergence-freeness hypothesis, its greatest post-fixed point.

In [7], in order to prove that a system of equations has a unique solution, we need to extend the transitions to *equation expressions* (i.e., contexts). For the same reasons, here we consider that contexts or equations may perform transitions, obtained as for the LTS, and assuming the hole does not perform any action. This is extended to traces. Hence, if $E\langle\widetilde{a}\rangle$ has the trace s (written $E\langle\widetilde{a}\rangle \stackrel{s}{\Longrightarrow}$), then for any $\widetilde{F}$, $E[\widetilde{F}]\langle\widetilde{a}\rangle \stackrel{s}{\Longrightarrow}$. For more details we refer the reader to [7].

Lemma 54 (Pre-fixed points, $\preceq_{\mathrm{tr}}$)**.** *Let $\mathcal{E}$ be a system of equations, and $\widetilde{\mathrm{K}}_{\mathcal{E}}$ its syntactic solution. If $\widetilde{F}$ is a pre-fixed point for $\preceq_{\mathrm{tr}}$ of $\mathcal{E}$, then $\widetilde{\mathrm{K}}_{\mathcal{E}} \preceq_{\mathrm{tr}} \widetilde{F}$.*

Proof. Consider a finite trace s of $\mathrm{K}_{\widetilde{E},i}\langle\widetilde{a}\rangle$. As it is finite, there must be an n such that s is a trace of $E_i^n\langle\widetilde{a}\rangle$, hence it is also a trace of $E_i^n[\widetilde{F}]\langle\widetilde{a}\rangle$. From $\widetilde{E}[\widetilde{F}] \preceq_{\mathrm{tr}} \widetilde{F}$, by congruence if follows that $E_i^{n+1}[\widetilde{F}] \preceq_{\mathrm{tr}} E_i^n[F_i]$, hence also $E_i^{n+1}[\widetilde{F}] \preceq_{\mathrm{tr}} F_i$. Hence, s is a trace of $F_i\langle\widetilde{a}\rangle$, and we can conclude by Proposition 53. $\square$

Lemma 55 (Post-fixed points, $\preceq_{\mathrm{tr}}$)**.** *Let $\mathcal{E}$ be a guarded system of equations, and $\widetilde{\mathrm{K}}_{\mathcal{E}}$ its syntactic solution. Suppose $\widetilde{\mathrm{K}}_{\mathcal{E}}$ has no divergences. If $\widetilde{F}$ is a post-fixed point for $\preceq_{\mathrm{tr}}$ of $\mathcal{E}$, then $\widetilde{F} \preceq_{\mathrm{tr}} \widetilde{\mathrm{K}}_{\mathcal{E}}$.*

The proof of Lemma 55 is similar to the proof of Theorem 16 (for bisimilarity). Details of the proof are given in Appendix E.

The following proof technique makes it possible to avoid referring to the syntactic solution of a system of equations, which is sometimes inconvenient.

Theorem 56. *Suppose that $\mathcal{E}$ is a guarded system of equations with a divergence-free syntactic solution. If $\widetilde{F}$ (resp. $\widetilde{G}$) is a pre-fixed point (resp. post-fixed point) for $\preceq_{\rm tr}$ of $\mathcal{E}$, then $\widetilde{G} \preceq_{\rm tr} \widetilde{F}$.*

Proof. Apply Lemma 55 to $\widetilde{F}$ and Lemma 54 to $\widetilde{G}$: this gives $\widetilde{G} \preceq_{\rm tr} \widetilde{\rm K}_{\mathcal{E}} \preceq_{\rm tr} \widetilde{F}$. □

We can also extend Lemma 18 to preorders. Given two systems of equations $\mathcal{E}$ and $\mathcal{E}'$, we say that $\mathcal{E}'$ *extends $\mathcal{E}$ with respect to a given preorder* if there exists a fixed set of indices J such that:

1. any pre-fixed point of $\mathcal{E}$ for the preorder can be obtained from a pre-fixed point of $\mathcal{E}'$ (for the same preorder) by removing the components corresponding to indices in J;
2. the same as (1) with post-fixed points in place of pre-fixed points.

Lemma 57. *Consider two systems of equations $\mathcal{E}'$ and $\mathcal{E}$ where $\mathcal{E}'$ extends $\mathcal{E}$ with respect to $\preceq_{\rm tr}$. Furthermore, suppose $\mathcal{E}'$ is guarded and has a divergence-free syntactic solution. If $\widetilde{F}$ is a pre-fixed point for $\preceq_{\rm tr}$ of $\mathcal{E}$, and $\widetilde{G}$ a post-fixed point for $\preceq_{\rm tr}$ of $\mathcal{E}$, then $\widetilde{G} \preceq_{\rm tr} \widetilde{F}$.*

Unique solution for trace equivalence. Theorem 56 gives the following property:

Corollary 58. *In $\mathrm{I}\pi$, a weakly guarded system of equations whose syntactic solution does not diverge has a unique solution for $\approx_{\rm tr}$.*

If $\widetilde{F} \approx_{\rm tr} \widetilde{E}[\widetilde{F}]$ and $\widetilde{G} \approx_{\rm tr} \widetilde{E}[\widetilde{G}]$, this gives, by applying Theorem 56 twice, $\widetilde{F} \preceq_{\rm tr} \widetilde{G}$ and $\widetilde{G} \preceq_{\rm tr} \widetilde{F}$, hence $\widetilde{F} \approx_{\rm tr} \widetilde{G}$.

6.3. Full Abstraction

The preorder on λ-terms induced by the contextual preorder is *η-eager normal-form similarity*, $\leq_\eta$. It is obtained by imposing that $M \leq_\eta N$ for all N, whenever M is divergent. Thus, with respect to the bisimilarity relation $\Leftrightarrow_\eta$, we only have to change clause (1) of Definition 3, by requiring only M to be divergent.

Definition 59 (η-eager normal-form similarity)**.** A relation $\mathcal{R}$ between λ-terms is an η-eager normal-form simulation if, whenever $M\mathcal{R}N$, one of the following holds:

1. M diverges
2. $M \Longrightarrow C_{\rm e}[xV]$ and $N \Longrightarrow C'_{\rm e}[xV']$ for some x, V, V', $C_{\rm e}$ and $C'_{\rm e}$ such that $V\mathcal{R}V'$ and $C_{\rm e}[z]\mathcal{R}C'_{\rm e}[z]$ for some z fresh in $C_{\rm e}, C'_{\rm e}$
3. $M \Longrightarrow \lambda x.\, M'$ and $N \Longrightarrow \lambda x.\, N'$ for some x, M', N' such that $M'\mathcal{R}N'$
4. $M \Longrightarrow x$ and $N \Longrightarrow x$ for some x
5. $M \Longrightarrow x$ and $N \Longrightarrow \lambda z.\, C_{\rm e}[xV]$ for some x, z, V and $C_{\rm e}$ such that $z\mathcal{R}V$ and $y\mathcal{R}C_{\rm e}[y]$ for some y fresh in $C_{\rm e}$
6. $N \Longrightarrow x$ and $M \Longrightarrow \lambda z.\, C_{\rm e}[xV]$ for some x, z, V and $C_{\rm e}$ such that $V\mathcal{R}z$ and $C_{\rm e}[y]\mathcal{R}y$ for some y fresh in $C_{\rm e}$

η-eager normal form similarity is the largest η-eager normal-form simulation.

Theorem 60 (Full abstraction on preorders)**.** *For any $M, N \in \Lambda$, we have $M \leq_\eta N$ iff $\mathcal{I}[\![M]\!] \lesssim_{\rm ct}^{\rm I\pi} \mathcal{I}[\![N]\!]$.*

The structure of the proofs is similar to that for bisimilarity, using however Theorem 56. We discuss the main aspects of the soundness and the completeness.

Soundness for trace inclusion. We show that $\mathcal{I}[\![M]\!] \preceq_{\rm tr} \mathcal{I}[\![N]\!]$ implies $M \leq_\eta N$. The proof follows the same lines as the proof from Section 4.3: we define the relation $\mathcal{R} := \{(M, N) \mid \mathcal{O}[\![M]\!] \preceq_{\rm tr} \mathcal{O}[\![N]\!]\}$, and show that it is an η-eager normal-form simulation. The proof carries over similarly, using the equivalents for trace inclusion of Lemmas 24, 28, 29 and 32 and Proposition 27.

Completeness for trace inclusion. Given an η-eager normal-form simulation $\mathcal{R}$, we define a system of equations $\mathcal{E}_{\mathcal{R}}$ as in Section 4.4. The only notable difference in the definition of the equations is in the case where $M\mathcal{R}N$, M diverges and N has an eager normal form. In this case, we use the following equation instead:

$$X_{M,N} = (\widetilde{y})\ \mathcal{I}[\![\Omega]\!]\ . \tag{5}$$

As in Section 4.4, we define a system of guarded equations $\mathcal{E}'_{\mathcal{R}}$ whose syntactic solutions do not diverge. Equation (5) is replaced with $X_{M,N} = (\widetilde{y}, p)\ \mathbf{0}$.

Exploiting Lemma 57, we can use unique solution for preorders (Theorem 56) with $\mathcal{E}_{\mathcal{R}}$ instead of $\mathcal{E}'_{\mathcal{R}}$.

Defining $\mathcal{I}^{c}[\![\mathcal{R}_1]\!]$ and $\mathcal{I}^{c}[\![\mathcal{R}_2]\!]$ as previously, we need to prove that $\mathcal{I}^{c}[\![\mathcal{R}_1]\!] \preceq_{\mathrm{tr}} \widetilde{E_{\mathcal{R}}}[\mathcal{I}^{c}[\![\mathcal{R}_1]\!]]$ and $\widetilde{E_{\mathcal{R}}}[\mathcal{I}^{c}[\![\mathcal{R}_2]\!]] \preceq_{\mathrm{tr}} \mathcal{I}^{c}[\![\mathcal{R}_2]\!]$. The former result is established along the lines of the analogous result in Section 4.4: indeed, $\mathcal{I}^{c}[\![\mathcal{R}_1]\!]$ is a solution of $\mathcal{E}_{\mathcal{R}}$ for $\approx$, and $\approx_{\mathrm{tr}}$ is coarser than $\approx$.

For the latter, the only difference is due to equation (5), when $M\mathcal{R}N$, and M diverges but not N. In that case, we have to prove that $\mathcal{I}[\![\Omega]\!] \preceq_{\mathrm{tr}} \mathcal{I}[\![N]\!]$, which follows easily because the only trace of $\mathcal{I}[\![\Omega]\!]$ is the empty one, hence $\mathcal{I}[\![\Omega]\!]\langle p\rangle \preceq_{\mathrm{tr}} P$ for any P.

We can then derive full abstraction for contextual equivalence as a corollary.

Corollary 61 (Full abstraction for $\simeq_{\mathrm{ct}}^{\mathrm{I}\pi}$)**.** *For any M, N in Λ, $M \Leftrightarrow_{\eta} N$ iff $\mathcal{I}[\![M]\!] \simeq_{\mathrm{ct}}^{\mathrm{I}\pi} \mathcal{I}[\![N]\!]$.*

7. Conclusion

In the paper we have studied the main question raised in Milner's landmark paper on functions as π-calculus processes, which is about the equivalence induced on λ-terms by their process encoding. We have focused on call-by-value, where the problem was still open; as behavioural equivalence on π-calculus we have taken contextual equivalence and barbed congruence (the most common 'linear' and 'branching' equivalences).

First we have shown that some expected equalities for open terms fail under Milner's encoding. We have considered two ways for overcoming this issue: rectifying the encodings (precisely, avoiding free outputs); restricting the target language to ALπ, so to control the capabilities of exported names. We have proved that, in both cases, the equivalence induced is Eager-Tree equality, modulo η (i.e., Lassen's η-eager normal-form bisimulation).

We have then introduced a preorder on these trees, and derived similar full abstraction results for them with respect to the contextual preorder on π-terms. The paper is also a test case for the technique of unique solution of equations (and inequations), which is essential in all our completeness proofs.

Lassen had introduced Eager Trees as the call-by-value analogous of Levy-Longo and Böhm Trees. The results in the paper confirm the claim, on process encodings of λ-terms: it was known that for (weak and strong) call-by-name, the equalities induced are those of Levy-Longo Trees and Böhm Trees [31].

For controlling capabilities, we have used ALπ. Another possibility would have been to use a type system. In this case however, the technique of unique solution of equations needs to be extended to typed calculi. We leave this for future work.

We also leave for future work a thorough comparison between the technique of unique solution of equations and techniques based on enhancements of the bisimulation proof method (the "up-to" proof techniques), including if and how our completeness results can be derived using the latter techniques. (We recall that the "up-to" proof techniques are used in the completeness proofs with respect to Levy-Longo Trees and Böhm Trees for the *call-by-name* encodings. We have discussed the problems with call-by-value in Remark 44.) In any case, even if other solutions existed, for this specific problem the unique solution technique appears to provide an elegant and natural framework to carry out the proofs.

For our encodings we have used the polyadic π-calculus; Milner's original paper [18] used the monadic calculus (the polyadic π-calculus makes the encoding easier to read; it had not been introduced at the time of [18]). We believe that polyadicity does not affect the results in the paper (the possibility of autoconcurrency breaks full abstraction of the encoding of the polyadic π-calculus into the monadic one, but autoconcurrency does not appear in the encoding of λ-terms).

In the call-by-value strategy we have followed, the function is reduced before the argument in an application. Our results can be adapted to the case in which the argument runs first, changing the definition of evaluation contexts. The parallel call-by-value, in which function and argument can run in parallel (considered in [19]), appears more delicate, as we cannot rely on the usual notion of evaluation context.

Interpretations of λ-calculi into π-calculi appear related to game semantics [5, 11, 10]. In particular, for untyped call-by-name they both allow us to derive Böhm Trees and Levy-Longo Trees [12, 21]. In this respect, it would be interesting to see whether the relationship between π-calculus and Eager Trees studied in this paper could help to establish similar relationships in game semantics.

References

[1] Samson Abramsky. The lazy λ-calculus. In D. Turner, editor, *Research Topics in Functional Programming*, pages 65–117. Addison Wesley, 1987.

[2] Beniamino Accattoli and Claudio Sacerdoti Coen. On the relative usefulness of fireballs. In *30th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2015, Kyoto, Japan, July 6-10, 2015*, pages 141–155, 2015.

[3] Beniamino Accattoli and Giulio Guerrieri. Open call-by-value. In *Proc. of APLAS 2016*, volume 10017 of *Lecture Notes in Computer Science*, pages 206–226. Springer Verlag, 2016.

[4] H.P. Barendregt. *The lambda calculus: its syntax and semantics*. Studies in logic and the foundations of mathematics. North-Holland, 1984.

[5] Martin Berger, Kohei Honda, and Nobuko Yoshida. Sequentiality and the pi-calculus. In *Proceedings of TLCA*, volume 2044 of *Lecture Notes in Computer Science*, pages 29–45. Springer, 2001.

[6] Romain Demangeon, Daniel Hirschkoff, and Davide Sangiorgi. Termination in impure concurrent languages. In *Proc. 21th Conf. on Concurrency Theory*, volume 6269 of *Lecture Notes in Computer Science*, pages 328–342. Springer, 2010.

[7] Adrien Durier, Daniel Hirschkoff, and Davide Sangiorgi. Divergence and Unique Solution of Equations. In *Proceedings of CONCUR 2017*, volume 85 of *LIPIcs*, pages 11:1–11:16. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2017.

[8] Adrien Durier, Daniel Hirschkoff, and Davide Sangiorgi. Eager functions as processes. In Anuj Dawar and Erich Grädel, editors, *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2018, Oxford, UK, July 09-12, 2018*, pages 364–373. ACM, 2018. `doi:10.1145/3209108.3209152`.

[9] J. Roger Hindley and Jonathan P. Seldin. *Introduction to Combinators and Lambda-Calculus*. Cambridge University Press, 1986.

[10] Kohei Honda and Nobuko Yoshida. Game-theoretic analysis of call-by-value computation. *Theor. Comput. Sci.*, 221(1-2):393–456, 1999.

[11] J. M. E. Hyland and C.-H. Luke Ong. Pi-calculus, dialogue games and PCF. In *Proceedings of FPCA 1995*, pages 96–107. ACM, 1995.

[12] Andrew D. Ker, Hanno Nickau, and C.-H. Luke Ong. Adapting innocent game models for the böhm treelambda-theory. *Theor. Comput. Sci.*, 308(1-3):333–366, 2003.

[13] Søren B. Lassen. Eager normal form bisimulation. In *20th IEEE Symposium on Logic in Computer Science (LICS 2005), 26-29 June 2005, Chicago, IL, USA, Proceedings*, pages 345–354. IEEE Computer Society, 2005.

[14] Søren B. Lassen and Paul Blain Levy. Typed normal form bisimulation. In *Proc. of Computer Science Logic CSL 2007*, volume 4646 of *Lecture Notes in Computer Science*, pages 283–297. Springer, 2007.

[15] Jean-Jacques Lévy. An algebraic interpretation of the lambda beta-calculus and a labeled lambda-calculus. In *Lambda-Calculus and Computer Science Theory, Proceedings of the Symposium Held in Rome, March 25-27, 1975*, volume 37 of *Lecture Notes in Computer Science*, pages 147–165. Springer, 1975.

[16] Giuseppe Longo. Set-theoretical models of lambda-calculus: theories, expansions, isomorphisms. *Annals of Pure and Applied Logic*, 24(2):153 – 188, 1983.

[17] Massimo Merro and Davide Sangiorgi. On asynchrony in name-passing calculi. *Mathematical Structures in Computer Science*, 14(5):715–767, 2004.

[18] Robin Milner. Functions as processes. Research Report RR-1154, INRIA, 1990.

[19] Robin Milner. Functions as processes. *Mathematical Structures in Computer Science*, 2(2):119–141, 1992.

[20] Robin Milner. The polyadic π-calculus: a tutorial. In *Logic and algebra of specification*, volume 94 of *NATO ASI Series (Series F: Computer & Systems Sciences)*, pages 203–246. Springer, 1993.

[21] C.-H. Luke Ong and Pietro Di Gianantonio. Games characterizing levy-longo trees. *Theor. Comput. Sci.*, 312(1):121–142, 2004.

[22] Benjamin C. Pierce and Davide Sangiorgi. Typing and subtyping for mobile processes. *Mathematical Structures in Computer Science*, 6(5):409–453, 1996.

[23] Gordon D. Plotkin. Call-by-name, call-by-value and the lambda-calculus. *Theor. Comput. Sci.*, 1(2):125–159, 1975.

[24] Simona Ronchi Della Rocca and Luca Paolini. *The Parametric Lambda Calculus - A Metamodel for Computation.* Texts in Theoretical Computer Science. An EATCS Series. Springer, 2004.

[25] Davide Sangiorgi. *Expressing mobility in process algebras : first-order and higher-order paradigms.* PhD thesis, University of Edinburgh, UK, 1993.

[26] Davide Sangiorgi. An investigation into functions as processes. In *Proc. of MFPS'93*, volume 802 of *Lecture Notes in Computer Science*, pages 143–159. Springer, 1993.

[27] Davide Sangiorgi. π-calculus, internal mobility, and agent-passing calculi. *Theor. Comput. Sci.*, 167(1&2):235–274, 1996.

[28] Davide Sangiorgi. Lazy functions and mobile processes. In *Proof, Language, and Interaction, Essays in Honour of Robin Milner*, pages 691–720. The MIT Press, 2000.

[29] Davide Sangiorgi. Termination of processes. *Mathematical Structures in Computer Science*, 16(1):1–39, 2006.

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

[31] Davide Sangiorgi and Xian Xu. Trees from functions as processes. In *Proceedings of CONCUR 2014*, volume 8704 of *Lecture Notes in Computer Science*, pages 78–92. Springer, 2014.

[32] Kristian Støvring and Søren B. Lassen. A complete, co-inductive syntactic theory of sequential control and state. In *Semantics and Algebraic Specification, Essays Dedicated to Peter D. Mosses on the Occasion of His 60th Birthday*, volume 5700 of *Lecture Notes in Computer Science*, pages 329–375. Springer, 2009.

[33] Bernardo Toninho and Nobuko Yoshida. On polymorphic sessions and functions - A tale of two (fully abstract) encodings. In *Proc. of ESOP 2018*, volume 10801 of *Lecture Notes in Computer Science*, pages 827–855. Springer, 2018.

[34] Nobuko Yoshida, Martin Berger, and Kohei Honda. Strong normalisation in the pi -calculus. *Inf. Comput.*, 191(2):145–202, 2004.

Appendix A. List of symbols for behavioural relations

The following table summarises the notation used for the equivalences and preorders in the paper.

$\Leftrightarrow$	eager normal-form bisimilarity	(Definition 3)
$\Leftrightarrow_\eta$	η-eager normal-form bisimilarity	(Definition 6)
$\leq_\eta$	η-eager normal-form similarity	(Section 6.3)
$\simeq^{\mathcal{L}}$	barbed congruence in $\mathcal{L}$	(Definition 8)
$\approx$	(weak) bisimilarity	(Definition 10)
$\preceq$	expansion	(Definition 11)
$\simeq^{\Lambda}_{\mathrm{ct}}$	contextual equivalence in Λ	(Definition 2)
$\simeq^{\mathcal{L}}_{\mathrm{ct}}$	contextual equivalence in $\mathcal{L}$	(Section 6.1)
$\lesssim^{\mathrm{I}\pi}_{\mathrm{ct}}$	contextual preorder in Iπ	(Section 6.1)
$\approx_{\mathrm{tr}}$	trace equivalence	(Section 6.1)
$\preceq_{\mathrm{tr}}$	trace inclusion	(Section 6.1)

where $\mathcal{L}$ is supposed to be a subcalculus of π; in the paper we have considered Iπ and ALπ.

Appendix B. Properties of Milner's encoding (Section 3)

Proposition 20 (Non-law)**.** *For any value V, we have:*

$$\mathcal{V}[\![I(xV)]\!] \not\simeq^{\pi} \mathcal{V}[\![xV]\!] \quad \text{and} \quad \mathcal{V}'[\![I(xV)]\!] \not\simeq^{\pi} \mathcal{V}'[\![xV]\!] \ .$$

Proof. For simplicity, we give the proof when $V = y$, and for encoding $\mathcal{V}$. The same can be shown for an arbitrary value V and for the encoding $\mathcal{V}'$, through similar calculations. We use algebraic laws of the equivalence $\simeq^{\pi}$, or of its associated proof techniques, to carry out the calculations (cf. [30]).

$$\begin{aligned}
\mathcal{V}[\![xy]\!]\langle p\rangle &= \boldsymbol{\nu} q\, (\overline{q}\langle x\rangle \mid q(u).\, \boldsymbol{\nu} r\, (\overline{r}\langle y\rangle \mid r(w).\, \overline{u}\langle w,p\rangle))\\
&\simeq^{\pi} \boldsymbol{\nu} r\, (\overline{r}\langle y\rangle \mid r(w).\, \overline{x}\langle w,p\rangle)\\
&\simeq^{\pi} \overline{x}\langle y,p\rangle\\
\mathcal{V}[\![I(xy)]\!]\langle p\rangle &\simeq^{\pi} \boldsymbol{\nu} q\, (\mathcal{V}[\![I]\!]\langle q\rangle \mid q(u).\, \boldsymbol{\nu} r\, (\mathcal{V}[\![xy]\!]\langle r\rangle \mid r(w).\, \overline{u}\langle w,p\rangle))\\
&\simeq^{\pi} \boldsymbol{\nu} q\, (\overline{q}(u).\, !u(z,q').\, \overline{q'}\langle z\rangle \mid q(u).\, \boldsymbol{\nu} r\, (\mathcal{V}[\![xy]\!]\langle r\rangle \mid r(w).\, \overline{u}\langle w,p\rangle))\\
&\simeq^{\pi} \boldsymbol{\nu} u\, (!u(z,q).\, \overline{q}\langle z\rangle \mid \boldsymbol{\nu} r\, (\overline{x}\langle y,r\rangle \mid r(w).\, \overline{u}\langle w,p\rangle))\\
&\simeq^{\pi} \boldsymbol{\nu} r\, (\overline{x}\langle y,r\rangle \mid r(w).\, \boldsymbol{\nu} u\, (!u(z,q).\, \overline{q}\langle z\rangle \mid \overline{u}\langle w,p\rangle))\\
&\simeq^{\pi} \boldsymbol{\nu} r\, (\overline{x}\langle y,r\rangle \mid r(w).\, \overline{p}\langle w\rangle)\\
&= \boldsymbol{\nu} q\, (\overline{x}\langle y,q\rangle \mid q(z).\, \overline{p}\langle z\rangle)\\
&= \boldsymbol{\nu} q\, (\overline{x}\langle y,q\rangle.\, q \triangleright p)
\end{aligned}$$

□

Appendix C. Soundness proof (Section 4)

Appendix C.1. Properties of the optimised encoding

Lemma 22. *We have:*

1. $\boldsymbol{\nu} q\, (p \triangleright q \mid q \triangleright r) \succeq p \triangleright r$*, for all continuation names* p, r*.*
2. $\boldsymbol{\nu} y\, (x \triangleright y \mid y \triangleright z) \succeq x \triangleright z$*, for all value names* x, z*.*

Proof. We first show the two following laws:

$$\begin{aligned}
\boldsymbol{\nu} q\, (p \triangleright q \mid q \triangleright r) &\sim p(x).\, \boldsymbol{\nu} q\, (\overline{q}(y).\, y \triangleright x \mid q \triangleright r)\\
&\succeq p(x).\, \boldsymbol{\nu} y\, (y \triangleright x \mid \overline{r}(z).\, z \triangleright y)\\
&\sim p(x).\, \overline{r}(z).\, \boldsymbol{\nu} y\, (z \triangleright y \mid y \triangleright x)
\end{aligned}$$

and

$$\begin{aligned}\boldsymbol{\nu}y\,(x \triangleright y \mid y \triangleright z) &\sim\, !x(p,x').\,\boldsymbol{\nu}y\,(\overline{y}(q,y').\,(y' \triangleright x' \mid q \triangleright p) \mid y(q,y').\,\overline{z}(r,z').\,(z' \triangleright y' \mid r \triangleright q))\\ &\succeq\, !x(p,x').\,\boldsymbol{\nu}q,y'\,.\,(y' \triangleright x' \mid q \triangleright p \mid \overline{z}(r,z').\,(z' \triangleright y' \mid r \triangleright q))\\ &\sim\, !x(p,x').\,\overline{z}(r,z').\,(\boldsymbol{\nu}q\,(r \triangleright q \mid q \triangleright p) \mid \boldsymbol{\nu}y'\,(z' \triangleright y' \mid y' \triangleright x'))\end{aligned}$$

We define a relation $\mathcal{R}$ as the relation that contains, for all continuation names p, r and for all value names x, z, the following pairs:

1. $\boldsymbol{\nu}q\,(p \triangleright q \mid q \triangleright r)$ and $p \triangleright r$
   $\boldsymbol{\nu}y\,(x \triangleright y \mid y \triangleright z)$ and $x \triangleright z$
2. $\overline{r}(z).\,\boldsymbol{\nu}y\,(z \triangleright y \mid y \triangleright x)$ and $\overline{r}(z).\,z \triangleright x$
   $\overline{z}(r,z').\,(\boldsymbol{\nu}q\,(r \triangleright q \mid q \triangleright p) \mid \boldsymbol{\nu}y'\,(z' \triangleright y' \mid y' \triangleright x'))$ and $\overline{z}(r,z').\,(r \triangleright p \mid z' \triangleright x')$
   (for all non-continuation names x or x', z')

We show that this is an expansion up to expansion and contexts, using the previous laws (each of those processes has only one possible action). □

In the proofs below, we shall use the following properties about the optimised encoding, without referring explicitly to this lemma.

Lemma 62.

- *For any M, p, $\mathcal{O}[\![M]\!]\langle p\rangle$ cannot perform any input at p.*
- *For any V, y, the only transition that $\mathcal{O}_{\mathrm{V}}[\![V]\!]\langle y\rangle$ can do is an input at y.*
- *For any M, x, p, if $x \in \mathsf{fv}(M)$, then x appears in $\mathcal{O}[\![M]\!]\langle p\rangle$ only in output subject position.*

The properties above follow by an easy induction. The last one allows us to use the distributivity properties of private replications [30] when reasoning algebraically about the encoding of λ-terms.

Lemma 23. *We have:*

1. $\boldsymbol{\nu}x\,(\mathcal{O}[\![M]\!]\langle p\rangle \mid x \triangleright y) \succeq \mathcal{O}[\![M\{y/x\}]\!]\langle p\rangle$.
2. $\boldsymbol{\nu}p\,(\mathcal{O}[\![M]\!]\langle p\rangle \mid p \triangleright q) \succeq \mathcal{O}[\![M]\!]\langle q\rangle$.
3. $\boldsymbol{\nu}y\,(\mathcal{O}_{\mathrm{V}}[\![V]\!]\langle y\rangle \mid x \triangleright y) \succeq \mathcal{O}_{\mathrm{V}}[\![V]\!]\langle x\rangle$.

Proof. We establish the conjunction of the three following properties:

(L1) $\boldsymbol{\nu}x\,(\mathcal{O}[\![M]\!]\langle p\rangle \mid x \triangleright y) \succeq \mathcal{O}[\![M\{y/x\}]\!]\langle p\rangle$, and, if M is equal to some value V, we have $\boldsymbol{\nu}x\,(\mathcal{O}_{\mathrm{V}}[\![V]\!]\langle y_1\rangle \mid x \triangleright y) \succeq \mathcal{O}_{\mathrm{V}}[\![V\{y/x\}]\!]\langle y_1\rangle$, for any y_1.

(L2) $\boldsymbol{\nu}p\,(\mathcal{O}[\![M]\!]\langle p\rangle \mid p \triangleright q) \succeq \mathcal{O}[\![M]\!]\langle q\rangle$.

(L3) If M is equal to some value V, $\boldsymbol{\nu}y\,(\mathcal{O}_{\mathrm{V}}[\![V]\!]\langle y\rangle \mid x \triangleright y) \succeq \mathcal{O}_{\mathrm{V}}[\![V]\!]\langle x\rangle$.

Indeed, as we show below, there are dependencies between these three properties, which prevent us from treating them separately.

We reason by induction over M, and introduce some notations. In each case, we use (IH1) to refer to property (L1) of the induction hypothesis, and similarly for (IH2) and (IH3).

To reason about (L1), we set $\mathsf{lhs} = \boldsymbol{\nu}x\,(\mathcal{O}[\![M]\!]\langle p\rangle \mid x \triangleright y)$ and $\mathsf{rhs} = \mathcal{O}[\![M\{y/x\}]\!]\langle p\rangle$.

First case: $M = z$**.**
(L1). Then $\mathsf{lhs} = \boldsymbol{\nu}x\,(\mathcal{O}[\![M]\!]\langle p\rangle \mid x \triangleright y) = \boldsymbol{\nu}x\,(\overline{p}(y_1).\,y_1 \triangleright z \mid x \triangleright y)$. There are two sub-cases.

- $M = z \neq x$. Then $\mathsf{lhs} \sim \overline{p}(y_1).\,y_1 \triangleright z = \mathsf{rhs}$ because $\boldsymbol{\nu}x\,x \triangleright y \sim \mathbf{0}$.

  Since M is a value, we must also check that $\boldsymbol{\nu}x\,(\mathcal{O}_{\mathrm{V}}[\![z]\!]\langle y_1\rangle \mid x \triangleright y) \succeq \mathcal{O}_{\mathrm{V}}[\![z\{y/x\}]\!]\langle y_1\rangle$. Lemma 22 allows us to show this.

- $M = x$. Then

$$\mathsf{lhs} \sim \overline{p}(y_1).\, \boldsymbol{\nu} x\, (y_1 \triangleright x \mid x \triangleright y) \tag{C.1}$$
$$\succeq \overline{p}(y_1).\, y_1 \triangleright y = \mathsf{rhs} \tag{C.2}$$

(C.1) holds because the only transition lhs can do is the output at p. (C.2) follows from Lemma 22.

Again, since x is a value, we have to show the corresponding property, which amounts to show $\boldsymbol{\nu} x\, (y_1 \triangleright x \mid x \triangleright y) \succeq y_1 \triangleright y$, which is given by Lemma 22.

(L2). We write in general, for any value V (the following reasoning is also used below, in the case where M is an abstraction):

$$\begin{aligned}
\boldsymbol{\nu} p\, (\mathcal{O}[\![V]\!]\langle p\rangle \mid p \triangleright q) &= \boldsymbol{\nu} p\, (\overline{p}(y).\, \mathcal{O}_{\mathrm{V}}[\![V]\!]\langle y\rangle \mid p \triangleright q) \\
&\succeq \boldsymbol{\nu} y\, (\mathcal{O}_{\mathrm{V}}[\![V]\!]\langle y\rangle \mid \overline{q}(y').\, y' \triangleright y) && \text{(C.3)} \\
&\sim \overline{q}(y').\, \boldsymbol{\nu} y\, (\mathcal{O}_{\mathrm{V}}[\![V]\!]\langle y\rangle \mid y' \triangleright y) && \text{(C.4)}
\end{aligned}$$

Step (C.3) holds because the first process deterministically reduces to the second, and step (C.4) holds because the only action the process can do is the output at q.

Now since $V = z$, we have $\mathcal{O}_{\mathrm{V}}[\![z]\!]\langle y\rangle = y \triangleright z$, and we obtain $\overline{q}(y').\, \boldsymbol{\nu} y\, (y \triangleright z \mid y' \triangleright y)$, a process that expands $\mathcal{I}[\![z]\!]\langle q\rangle$ by Lemma 22.

(L3). We check that we do have $\boldsymbol{\nu} y\, (y \triangleright z \mid x \triangleright y) \succeq x \triangleright z$ by Lemma 22.

Second case: $M = \lambda z.\, M'$.

(L1). We distinguish two cases.

- If $z \neq x$, then we can write

$$\begin{aligned}
\mathsf{lhs} &= \boldsymbol{\nu} x\, (\overline{p}(y_1).\, !y_1(z,q).\, \mathcal{O}[\![M']\!]\langle q\rangle \mid x \triangleright y) \\
&\sim \overline{p}(y_1).\, !y_1(z,q).\, \boldsymbol{\nu} x\, (\mathcal{O}[\![M']\!]\langle q\rangle \mid x \triangleright y) && \text{(C.5)} \\
&\succeq \mathsf{rhs} && \text{(C.6)}
\end{aligned}$$

Step (C.5) holds because x does not occur in the two prefixes, and step (C.6) holds by (IH1).

Since M is a value, we have to prove $\boldsymbol{\nu} x\, (\mathcal{O}_{\mathrm{V}}[\![\lambda z.\, M']\!]\langle y_1\rangle \mid x \triangleright y) \succeq \mathcal{O}_{\mathrm{V}}[\![\lambda z.\, M'\{y/x\}]\!]\langle y_1\rangle$. This follows by (IH1), along the lines of the above proof.

- If $M = \lambda x.\, M'$, then $x \notin \mathrm{fn}(\mathcal{O}[\![M]\!]\langle p\rangle)$ and we can observe that: first, the link $x \triangleright y$ together with the restriction on x can be erased up to $\sim$; second, $M\{y/x\} = M$. We can thus conclude.

Again, M is a value, so we need to prove the corresponding property, which is done as in the proof above.

(L2). We know from the first case in the induction (step (C.4)) that

$$\begin{aligned}
\boldsymbol{\nu} p\, (\mathcal{O}[\![\lambda z.\, M']\!]\langle p\rangle \mid p \triangleright q) &\succeq \overline{q}(y').\, \boldsymbol{\nu} y\, (\mathcal{O}_{\mathrm{V}}[\![\lambda z.\, M']\!]\langle y\rangle \mid y' \triangleright y) \\
&= \overline{q}(y').\, \boldsymbol{\nu} y\, (!y(z,q').\, \mathcal{O}[\![M']\!]\langle q'\rangle \mid y' \triangleright y) \\
&\sim \overline{q}(y').\, !y'(z_1,q_1).\, \boldsymbol{\nu} y\, (!y(z,q').\, \mathcal{O}[\![M']\!]\langle q'\rangle \\
&\qquad \mid \overline{y}(z,q').\, (z \triangleright z_1 \mid q' \triangleright q_1)) && \text{(C.7)} \\
&\succeq \overline{q}(y').\, !y'(z_1,q_1).\, (\boldsymbol{\nu} y\, , z, q')(!y(z,q').\, \mathcal{O}[\![M']\!]\langle q'\rangle \\
&\qquad \mid \mathcal{O}[\![M']\!]\langle q'\rangle \mid z \triangleright z_1 \mid q' \triangleright q_1) && \text{(C.8)} \\
&\sim \overline{q}(y').\, !y'(z_1,q_1).\, (\boldsymbol{\nu} z\, , q')(\mathcal{O}[\![M']\!]\langle q'\rangle \mid z \triangleright z_1 \mid q' \triangleright q_1) && \text{(C.9)} \\
&\succeq \overline{q}(y').\, !y'(z_1,q_1).\, \mathcal{O}[\![M'\{z_1/z\}]\!]\langle q_1\rangle && \text{(C.10)} \\
&= \mathcal{O}[\![\lambda z.\, M']\!]\langle q\rangle && \text{(C.11)}
\end{aligned}$$

Step (C.7) holds because the input at y' is the only transition that can be performed after the bound output at q. Step (C.8) holds by performing a deterministic communication on y.

Step (C.9) simply consists in garbage-collecting the input at y. For step (C.10), we use (IH1) and (IH2)— note that (IH2) is necessary here to establish (IH1).

(L3). We write

$$\begin{aligned}\boldsymbol{\nu}y\,(!y(z,q).\,\mathcal{O}[\![M']\!]\langle q\rangle \mid x \triangleright y) &\sim\; !x(z',q').\,\boldsymbol{\nu}y\,(!y(z,q).\,\mathcal{O}[\![M']\!]\langle q\rangle \\ &\qquad \mid \overline{y}(z,q).\,(z \triangleright z' \mid q \triangleright q')) \\ &\succeq\; !x(z',q').\,(\boldsymbol{\nu}z\,,q)(\mathcal{O}[\![M']\!]\langle q\rangle \mid z \triangleright z' \mid q \triangleright q') \\ &\succeq\; !x(z',q').\,\mathcal{O}[\![M'\{z'/z\}]\!]\langle q'\rangle\end{aligned}$$

The reasoning steps are like in the proof above: expand the behaviour of a link process, perform a deterministic communication, and rely on (IH1) and (IH2) to get rid of the forwarders. We note that (IH1) and (IH2) are used to prove (L3) in this case.

Third case: M is an application.

(L3). We do not have to consider (L3) in this case, since M is not a value.

There are 5 sub-cases, according to the definition of the optimised encoding of Figure 3.

We let W stand for the process $\overline{y_1}(w',r').\,(w' \triangleright w \mid r' \triangleright p)$, which is used in four of the clauses in the encoding for application.

(L2). To prove (L2), we reason in the same way in four sub-cases, namely all except $M = z_1V$: in these sub-cases the only occurrence of p in $\mathcal{O}[\![M]\!]\langle p\rangle$ is in the sub-process W. We reason modulo strong bisimilarity ($\sim$) to bring the forwarder $p \triangleright q$ close to that occurrence, yielding a subterm of the form $\overline{y_1}(w',r').\,(w' \triangleright w \mid \boldsymbol{\nu}p\,(r' \triangleright p \mid p \triangleright q))$. We use Lemma 22 to deduce that this process expands $\overline{y_1}(w',r').\,(w' \triangleright w \mid r' \triangleright q)$, which allows us to establish (L2).

Similarly, in the last case ($M = z_1V$), we write

$$\boldsymbol{\nu}p\,(\overline{z_1}(z,q').\,(\mathcal{O}_{\mathrm{V}}[\![V]\!]\langle z\rangle \mid q' \triangleright p) \mid p \triangleright q) \;\sim\; \overline{z_1}(z,q').\,(\mathcal{O}_{\mathrm{V}}[\![V]\!]\langle z\rangle \mid \boldsymbol{\nu}p\,(q' \triangleright p \mid p \triangleright q)),$$

and we conclude using Lemma 22.

(L1). We analyse the 5 cases corresponding to the optimised encoding of application.

- $M = M'N'$, and none of M' and N' are values.

  Then

  $$\begin{aligned}\mathsf{lhs} &= \boldsymbol{\nu}x\,(\boldsymbol{\nu}q\,(\mathcal{O}[\![M']\!]\langle q\rangle \mid q(y_1).\,\boldsymbol{\nu}r\,(\mathcal{O}[\![N']\!]\langle r\rangle \mid r(w).\,\mathsf{W})) \mid x \triangleright y) \\ &\sim \boldsymbol{\nu}q\,(\boldsymbol{\nu}x\,(\mathcal{O}[\![M']\!]\langle q\rangle \mid x \triangleright y) \mid q(y_1).\,\boldsymbol{\nu}r\,(\boldsymbol{\nu}x\,(\mathcal{O}[\![N']\!]\langle r\rangle \mid x \triangleright y) \mid r(w).\,\mathsf{W})) \quad \text{(C.12)} \\ &\succeq \boldsymbol{\nu}q\,(\mathcal{O}[\![M'\{y/x\}]\!]\langle q\rangle \mid q(y_1).\,\boldsymbol{\nu}r\,(\mathcal{O}[\![N'\{y/x\}]\!]\langle r\rangle \mid r(w).\,\mathsf{W})) = \mathsf{rhs} \quad \text{(C.13)}\end{aligned}$$

  Step (C.12) holds by distributivity properties of private replications, and step (C.13) holds by applying (IH1) twice.

- $M = M'V$. Then

  $$\begin{aligned}\mathsf{lhs} &= \boldsymbol{\nu}x\,(\boldsymbol{\nu}q\,(\mathcal{O}[\![M']\!]\langle q\rangle \mid q(y_1).\,\boldsymbol{\nu}w\,(\mathcal{O}_{\mathrm{V}}[\![V]\!]\langle w\rangle \mid \mathsf{W})) \mid x \triangleright y) \\ &\sim \boldsymbol{\nu}q\,(\boldsymbol{\nu}x\,(\mathcal{O}[\![M']\!]\langle q\rangle \mid x \triangleright y) \mid q(y_1).\,\boldsymbol{\nu}w\,(\boldsymbol{\nu}x\,(\mathcal{O}_{\mathrm{V}}[\![V]\!]\langle w\rangle \mid x \triangleright y) \mid \mathsf{W}) \quad \text{(C.14)} \\ &\succeq \mathsf{rhs} \quad \text{(C.15)}\end{aligned}$$

  Step (C.14) is proved using the distributivity properties of private replications; step (C.15) follows by (IH1).

- $M = VM'$. This case is proved using the same kind of reasoning as the previous one.

- $M = (\lambda z.\,M')V$. Then

  $$\begin{aligned}\mathsf{lhs} &= \boldsymbol{\nu}x\,(\boldsymbol{\nu}y_1, w(\mathcal{O}_{\mathrm{V}}[\![\lambda z.\,M']\!]\langle y_1\rangle \mid \mathcal{O}_{\mathrm{V}}[\![V]\!]\langle w\rangle \mid \mathsf{W}) \mid x \triangleright y) \\ &\sim \boldsymbol{\nu}y_1, w(\boldsymbol{\nu}x\,(\mathcal{O}_{\mathrm{V}}[\![\lambda z.\,M']\!]\langle y_1\rangle \mid x \triangleright y) \mid \boldsymbol{\nu}x\,(\mathcal{O}_{\mathrm{V}}[\![V]\!]\langle w\rangle \mid x \triangleright y) \mid \mathsf{W}) \\ &\succeq \mathsf{rhs}\end{aligned}$$

  Here again, we distribute the forwarder and then apply (IH1) twice (for the encoding $\mathcal{O}_{\mathrm{V}}[\![\cdot]\!]\langle\cdot\rangle$).

- $M = z_1 V$. We have

$$\mathsf{lhs} \;=\; \boldsymbol{\nu} x\, (\overline{z_1}(z,q).\,(\mathcal{O}_{\mathrm{V}}[\![V]\!]\langle z\rangle \mid q \triangleright p) \mid x \triangleright y)$$

We distinguish two cases:

  - if $z_1 \neq x$, $z_1\{y/x\} = z_1$, and

$$\begin{aligned}\mathsf{lhs} &\sim \overline{z_1}(z,q).\,\boldsymbol{\nu} x\,(\mathcal{O}_{\mathrm{V}}[\![V]\!]\langle z\rangle \mid x \triangleright y \mid q \triangleright p)\\ &\succeq \overline{z_1}(z,q).\,(\mathcal{O}_{\mathrm{V}}[\![V\{y/x\}]\!]\langle z\rangle \mid q \triangleright p) = \mathsf{rhs}\end{aligned}$$

    Above, we apply (IH1) for $\mathcal{O}_{\mathrm{V}}[\![V]\!]\langle z\rangle$.

  - if $z_1 = x$, then the only transition lhs can make is a communication at x, so

$$\begin{aligned}\mathsf{lhs} &\succeq (\boldsymbol{\nu} x\,,q,z)(\mathcal{O}_{\mathrm{V}}[\![V]\!]\langle z\rangle \mid q \triangleright p \mid x \triangleright y \mid \overline{y}(w_1,q_1).\,(q_1 \triangleright q \mid w_1 \triangleright z))\\ &\sim \overline{y}(w_1,q_1).\,\boldsymbol{\nu} x\,(\boldsymbol{\nu} z\,(\mathcal{O}_{\mathrm{V}}[\![V]\!]\langle z\rangle \mid w_1 \triangleright z) \mid \boldsymbol{\nu} q\,(q_1 \triangleright q \mid q \triangleright p) \mid x \triangleright y) \qquad \text{(C.16)}\end{aligned}$$

    Step (C.16) holds because the only transition the process can make is the bound output at y.
    We then use Lemma 22 to contract the forwarders, yielding $q_1 \triangleright p$; we use (IH3) to erase the forwarder $w_1 \triangleright z$, and we use (IH1) to replace $\boldsymbol{\nu} x\,(\mathcal{O}_{\mathrm{V}}[\![V]\!]\langle w_1\rangle \mid x \triangleright y)$ with $\mathcal{O}_{\mathrm{V}}[\![V\{y/x\}]\!]\langle w_1\rangle$. This finally yields rhs.
    We remark here that we use (IH3) to show (L1).

□

Lemma 24. *$\mathcal{I}[\![M]\!] \succeq \mathcal{O}[\![M]\!]$, for all $M \in \Lambda$.*

Proof. We reason by induction over M.
Case $M = x$**.** By definition, $\mathcal{O}[\![M]\!]\langle p\rangle = \mathcal{I}[\![M]\!]\langle p\rangle$.
Case $M = \lambda x.\,N$**.** Assuming $\mathcal{O}[\![N]\!]\langle q\rangle \succeq \mathcal{I}[\![N]\!]\langle q\rangle$, we have, by definition

$$\begin{aligned}\mathcal{O}[\![M]\!]\langle p\rangle &= \overline{p}(y).\,!y(x,q).\,\mathcal{O}[\![N]\!]\langle q\rangle\\ &\succeq \overline{p}(y).\,!y(x,q).\,\mathcal{I}[\![N]\!]\langle q\rangle \;= \mathcal{I}[\![M]\!]\langle p\rangle\end{aligned}$$

Case $M = M_1 M_2$**.** We have by definition

$$\begin{aligned}\mathcal{I}[\![M_1 M_2]\!]\langle p\rangle &= \boldsymbol{\nu} q\,(\mathcal{I}[\![M_1]\!]\langle q\rangle \mid q(y).\,\boldsymbol{\nu} r\,(\mathcal{I}[\![M_2]\!]\langle r\rangle \mid r(w).\,\mathsf{W}))\\ &\qquad \text{with } \mathsf{W} = \overline{y}(w',p').\,(w' \triangleright w \mid p' \triangleright p)\end{aligned}$$

We distinguish 5 cases, according to the definition of the optimised encoding in Figure 3.

- $M_1 = x, M_2 = V$.

$$\begin{aligned}\mathcal{I}[\![xV]\!]\langle p\rangle &\succeq \boldsymbol{\nu} q\,(\overline{q}(y).\,y \triangleright x \mid q(y).\,\boldsymbol{\nu} r\,(\mathcal{O}[\![V]\!]\langle r\rangle \mid r(w).\,\mathsf{W})) &\text{(C.17)}\\ &\succeq \boldsymbol{\nu} y,w\,(y \triangleright x \mid \mathcal{O}_{\mathrm{V}}[\![V]\!]\langle w\rangle \mid \overline{y}(w',p').\,(w' \triangleright w \mid p' \triangleright p))\\ &\succeq \boldsymbol{\nu} w,w'\,(\overline{x}(z,q).\,(z \triangleright w' \mid q \triangleright p') \mid \mathcal{O}_{\mathrm{V}}[\![V]\!]\langle w\rangle \mid w' \triangleright w \mid p' \triangleright p)\\ &\sim \overline{x}(z,q).\,\boldsymbol{\nu} w,w'\,(\mathcal{O}_{\mathrm{V}}[\![V]\!]\langle w\rangle \mid z \triangleright w' \mid w' \triangleright w \mid q \triangleright p' \mid p' \triangleright p) &\text{(C.18)}\\ &\succeq \overline{x}(z,q).\,\boldsymbol{\nu} w\,(\mathcal{O}_{\mathrm{V}}[\![V]\!]\langle w\rangle \mid z \triangleright w \mid q \triangleright p) &\text{(C.19)}\\ &\succeq \overline{x}(z,q).\,(\mathcal{O}_{\mathrm{V}}[\![V]\!]\langle z\rangle \mid q \triangleright p) \;=\; \mathcal{O}[\![xV]\!]\langle p\rangle &\text{(C.20)}\end{aligned}$$

  Step (C.17) follows by definition (for the encoding of x), and by induction (for the encoding of V). The two following $\succeq$ steps are derived by performing deterministic τ transitions. We then remark that the only action that can be performed is a bound output at x (step (C.18)), and contract fowarders using Lemma 22 (step (C.19)). Finally, we use Lemma 23 in step (C.20).

- Case $M = (\lambda x.\, N)V$.

$$
\begin{array}{rcll}
\mathcal{I}[\![(\lambda x.\, N)V]\!]\langle p\rangle & = & \boldsymbol{\nu} q\, (\overline{q}(y).\, !y(x,p').\, \mathcal{I}[\![N]\!]\langle p'\rangle \mid q(y).\, \boldsymbol{\nu} r\, (\mathcal{I}[\![V]\!]\langle r\rangle \mid r(w).\, \mathsf{W}))) & \\
& \succeq & \boldsymbol{\nu} q\, (\overline{q}(y).\, !y(x,p').\, \mathcal{I}[\![N]\!]\langle p'\rangle & \\
& & \quad \mid q(y).\, \boldsymbol{\nu} r\, (\overline{r}(w).\, \mathcal{O}_{\mathrm{V}}[\![V]\!]\langle w\rangle \mid r(w).\, \mathsf{W})) & (C.21) \\
& \succeq & \boldsymbol{\nu}(y,w)\, (!y(x,p').\, \mathcal{I}[\![N]\!]\langle p'\rangle \mid \mathcal{O}_{\mathrm{V}}[\![V]\!]\langle w\rangle \mid \mathsf{W}) & (C.22) \\
& \succeq & \boldsymbol{\nu}(y,w)\, (!y(x,p').\, \mathcal{O}[\![N]\!]\langle p'\rangle \mid \mathcal{O}_{\mathrm{V}}[\![V]\!]\langle w\rangle \mid \mathsf{W}) & (C.23) \\
& = & \mathcal{O}[\![M]\!]\langle p\rangle &
\end{array}
$$

  Steps (C.21) and (C.23) follow by induction. Step (C.22) follows by performing two deterministic τ-transitions and garbage-collecting the restrictions on q and r.

- Case $M_1 = V, M_2 = N$.

$$
\begin{array}{rcl}
\mathcal{I}[\![VN]\!]\langle p\rangle & \succeq & \boldsymbol{\nu} q\, (\overline{q}(y).\, \mathcal{O}_{\mathrm{V}}[\![V]\!]\langle y\rangle \mid q(y).\, \boldsymbol{\nu} r\, (\mathcal{O}[\![N]\!]\langle r\rangle \mid r(w).\, \mathsf{W})) \\
& \succeq & \boldsymbol{\nu} y\, (\mathcal{O}_{\mathrm{V}}[\![V]\!]\langle y\rangle \mid \boldsymbol{\nu} r\, (\mathcal{O}[\![N]\!]\langle r\rangle \mid r(w).\, \mathsf{W})) \;= \mathcal{O}[\![VN]\!]\langle p\rangle
\end{array}
$$

  Again, we use the inductive hypothesis, and perform a deterministic τ-transition on q.

- Case $M_1 = N, M_2 = V$.

$$
\begin{array}{rcl}
\mathcal{I}[\![NV]\!]\langle p\rangle & \succeq & \boldsymbol{\nu} q\, (\mathcal{O}[\![N]\!]\langle q\rangle \mid q(y).\, \boldsymbol{\nu} r\, (\overline{r}(w).\, \mathcal{O}_{\mathrm{V}}[\![V]\!]\langle r\rangle \mid r(w).\, \mathsf{W})) \\
& \succeq & \boldsymbol{\nu} q\, (\mathcal{O}[\![N]\!]\langle q\rangle \mid q(y).\, \boldsymbol{\nu} w\, (\mathcal{O}_{\mathrm{V}}[\![V]\!]\langle w\rangle \mid \mathsf{W})) \;= \mathcal{O}[\![NV]\!]\langle p\rangle
\end{array}
$$

  Again, we first use the inductive hypothesis, then contract a deterministic communication on r.

- Finally, if neither M_1 nor M_2 is a value, the two encodings coincide, and the property is immediate.

□

Appendix C.2. Operational Correspondence and Soundness

The following lemma is the central property we need to derive the validity of β_{v}-reduction.

Lemma 63. $\boldsymbol{\nu} x\, (\mathcal{O}[\![M]\!]\langle p\rangle \mid \mathcal{O}_{\mathrm{V}}[\![V]\!]\langle x\rangle) \succeq \mathcal{O}[\![M\{V/x\}]\!]\langle p\rangle$.

Proof. We establish the following property

$$
\begin{array}{l}
\boldsymbol{\nu} x\, (\mathcal{O}[\![M]\!]\langle p\rangle \mid \mathcal{O}_{\mathrm{V}}[\![V]\!]\langle x\rangle) \succeq \mathcal{O}[\![M\{V/x\}]\!]\langle p\rangle \\
\text{and, if } M \text{ is a value } V',\ \boldsymbol{\nu} x\, (\mathcal{O}_{\mathrm{V}}[\![V']\!]\langle y\rangle \mid \mathcal{O}_{\mathrm{V}}[\![V]\!]\langle x\rangle \succeq \mathcal{O}_{\mathrm{V}}[\![V'\{V/x\}]\!]\langle y\rangle
\end{array}
$$

by induction over the size of M. We write rhs for the right hand side of the first relation above, that is, rhs stands for $\mathcal{O}[\![M\{V/x\}]\!]\langle p\rangle$. We use similarly lhs for $\boldsymbol{\nu} x\, (\mathcal{O}[\![M]\!]\langle p\rangle \mid \mathcal{O}_{\mathrm{V}}[\![V]\!]\langle x\rangle)$.

We distinguish several cases, following the definition of the optimised encoding in Figure 3.

- M **is a variable.** We distinguish two sub-cases.
  - $M = z, z \neq x$. Then $M\{V/x\} = z$, and we can write

$$
\begin{array}{rcll}
\mathsf{lhs} & = & \boldsymbol{\nu} x\, (\overline{p}(y).\, y \triangleright z \mid \mathcal{O}_{\mathrm{V}}[\![V]\!]\langle x\rangle) & \\
& \sim & \overline{p}(y).\, y \triangleright z & (C.24) \\
& = & \mathsf{rhs} &
\end{array}
$$

    Relation (C.24) above holds because x is fresh for $\overline{p}(y).\, y \triangleright z$ and because $\mathcal{O}_{\mathrm{V}}[\![V]\!]\langle x\rangle$ starts with an input on x.

    Since M is a value, we must also prove the second relation mentioned above. We have indeed $\boldsymbol{\nu} x\, (\mathcal{O}_{\mathrm{V}}[\![z]\!]\langle y\rangle \mid \mathcal{O}_{\mathrm{V}}[\![V]\!]\langle x\rangle) \sim y \triangleright z = \mathcal{O}_{\mathrm{V}}[\![V]\!]\langle x\rangle$, and we can conclude since $\sim \subseteq \succeq$.

– $M = x$. Then $M\{V/x\} = V$, and we can write

$$\begin{aligned}
\mathsf{lhs} &= \boldsymbol{\nu} x\,(\overline{p}(y).\,y \triangleright x \mid \mathcal{O}_{\mathrm{V}}[\![V]\!]\langle x\rangle) \\
&\sim \overline{p}(y).\,\boldsymbol{\nu} x\,(\mathcal{O}_{\mathrm{V}}[\![V]\!]\langle x\rangle \mid y \triangleright x) && \text{(C.25)} \\
&\succeq \overline{p}(y).\,\mathcal{O}_{\mathrm{V}}[\![V]\!]\langle y\rangle && \text{(C.26)} \\
&= \mathsf{rhs}
\end{aligned}$$

Relation (C.25) holds because $\mathcal{O}_{\mathrm{V}}[\![V]\!]\langle x\rangle$ starts with an input at x. Relation (C.26) follows by Lemma 23.

Again, in this case M is a value, so we must also prove the second relation. We can indeed check that $\boldsymbol{\nu} x\,(\mathcal{O}_{\mathrm{V}}[\![x]\!]\langle y\rangle \mid \mathcal{O}_{\mathrm{V}}[\![V]\!]\langle x\rangle) \succeq \mathcal{O}_{\mathrm{V}}[\![V]\!]\langle y\rangle$ by Lemma 23.

- M **is an abstraction.** We distinguish two sub-cases.

– $M = \lambda x.\,M'$. Then $M\{V/x\} = M$, and we can write

$$\begin{aligned}
\mathsf{lhs} &= \boldsymbol{\nu} x\,(\overline{p}(y).\,!y(x,q).\,\mathcal{O}[\![M']\!]\langle q\rangle \mid \mathcal{O}_{\mathrm{V}}[\![V]\!]\langle x\rangle) \\
&\sim \overline{p}(y).\,!y(x,q).\,\mathcal{O}[\![M']\!]\langle q\rangle && \text{(C.27)} \\
&= \mathsf{rhs}
\end{aligned}$$

Relation (C.27) above holds because x does not occur free in $\overline{p}(y).\,!y(x,q)\mathcal{O}[\![M']\!]\langle q\rangle$, and $\mathcal{O}_{\mathrm{V}}[\![V]\!]\langle x\rangle$ starts with an input at x.

M is a value, so we must prove the second relation as well. We have

$$\begin{aligned}
&\boldsymbol{\nu} x\,(\mathcal{O}_{\mathrm{V}}[\![\lambda x.\,M']\!]\langle y\rangle \mid \mathcal{O}_{\mathrm{V}}[\![V]\!]\langle x\rangle) \\
&= \boldsymbol{\nu} x\,(!y(x,q).\,\mathcal{O}[\![M']\!]\langle q\rangle \mid \mathcal{O}_{\mathrm{V}}[\![V]\!]\langle x\rangle) \\
&\sim\ !y(x,q).\,\mathcal{O}[\![M']\!]\langle q\rangle \mid \boldsymbol{\nu} x\,\mathcal{O}_{\mathrm{V}}[\![V]\!]\langle x\rangle && \text{(C.28)} \\
&\sim \mathcal{O}_{\mathrm{V}}[\![M]\!]\langle y\rangle && \text{(C.29)}
\end{aligned}$$

Relation (C.28) holds because x does not occur free in $!y(x,q).\,\mathcal{O}[\![M']\!]\langle q\rangle$, and relation (C.29) holds because the only possible transition of $\mathcal{O}_{\mathrm{V}}[\![V]\!]\langle x\rangle$ is an input at x.

– $M = \lambda z.\,M'$ with $z \neq x$. Then $M\{V/x\} = \lambda z.\,(M'\{V/x\})$, and by definition, $\mathsf{rhs} = \overline{p}(y).\,!y(z,q).\,\mathcal{O}[\![M'\{V/x\}]\!]\langle q\rangle$.

We can write

$$\begin{aligned}
\mathsf{lhs} &= \boldsymbol{\nu} x\,(\overline{p}(y).\,!y(z,q).\,\mathcal{O}[\![M']\!]\langle q\rangle \mid \mathcal{O}_{\mathrm{V}}[\![V]\!]\langle x\rangle) \\
&\sim \overline{p}(y).\,!y(z,q).\,\boldsymbol{\nu} x\,(\mathcal{O}[\![M']\!]\langle q\rangle \mid \mathcal{O}_{\mathrm{V}}[\![V]\!]\langle x\rangle) \\
&\succeq \overline{p}(y).\,!y(z,q).\,\mathcal{O}_{\mathrm{V}}[\![M'\{V/x\}]\!]\langle q\rangle = \mathsf{rhs}
\end{aligned}$$

M is a value, so we must also prove the second relation:

$$\begin{aligned}
&\boldsymbol{\nu} x\,(\mathcal{O}_{\mathrm{V}}[\![\lambda z.\,M']\!]\langle y\rangle \mid \mathcal{O}_{\mathrm{V}}[\![V]\!]\langle x\rangle) \\
&= \boldsymbol{\nu} x\,(!y(z,q).\,\mathcal{O}[\![M']\!]\langle q\rangle \mid \mathcal{O}_{\mathrm{V}}[\![V]\!]\langle x\rangle) \\
&\sim\ !y(z,q).\,\boldsymbol{\nu} x\,(\mathcal{O}[\![M']\!]\langle q\rangle \mid \mathcal{O}_{\mathrm{V}}[\![V]\!]\langle x\rangle) && \text{(C.30)} \\
&\succeq\ !y(z,q).\,\mathcal{O}[\![M'\{x/V\}]\!]\langle q\rangle && \text{(C.31)} \\
&= \mathcal{O}_{\mathrm{V}}[\![M\{x/V\}]\!]\langle y\rangle
\end{aligned}$$

- M **is an application.** We distinguish 5 sub-cases, according to the definition of the optimised encoding of Figure 3. In the following, we let W stand for the process $\overline{y}(w',r').\,(w' \triangleright w \mid r' \triangleright p)$, which is used in the different clauses in the encoding.

We also make use of some standard *properties of replicated resources* [30].

– $M = M'N'$, and none of M' and N' are values. Then we have:

$$\begin{aligned}
\mathsf{lhs} &= \boldsymbol{\nu} x\,\boldsymbol{\nu} q\,(\mathcal{O}[\![M]\!]\langle q\rangle \mid \mathcal{O}_{\mathrm{V}}[\![V]\!]\langle x\rangle \mid q(y).\,\boldsymbol{\nu} r\,(\mathcal{O}[\![N]\!]\langle r\rangle \mid r(w).\,\mathsf{W})) \\
&\sim \boldsymbol{\nu} q\,\boldsymbol{\nu} x\,(\mathcal{O}[\![M]\!]\langle q\rangle \mid \mathcal{O}_{\mathrm{V}}[\![V]\!]\langle x\rangle \\
&\qquad \mid q(y).\,\boldsymbol{\nu} r\,(\boldsymbol{\nu} x\,(\mathcal{O}[\![N]\!]\langle r\rangle \mid \mathcal{O}_{\mathrm{V}}[\![V]\!]\langle x\rangle) \mid r(w).\,\mathsf{W})) && \text{(C.32)} \\
&\succeq \boldsymbol{\nu} q\,(\mathcal{O}[\![M\{V/x\}]\!]\langle q\rangle \mid q(y).\,\boldsymbol{\nu} r\,(\mathcal{O}[\![N\{V/x\}]\!]\langle r\rangle \mid r(w).\,\mathsf{W})) && \text{(C.33)} \\
&= \mathsf{rhs}
\end{aligned}$$

Relation (C.32) holds by the distributivity properties of private replications [30] (note in particular that x is not used in input in $\mathcal{O}[\![M]\!]\langle q\rangle$ and in $\mathcal{O}[\![N]\!]\langle r\rangle$). Relation (C.33) holds by using the induction hypothesis twice.

– $M = M'V'$. Then

$$\begin{aligned}
\mathsf{lhs} &= \boldsymbol{\nu} x\,(\boldsymbol{\nu} q\,(\mathcal{O}[\![M']\!]\langle q\rangle \mid q(y).\,\boldsymbol{\nu} w\,(\mathcal{O}_{\mathrm{V}}[\![V']\!]\langle w\rangle \mid \mathsf{W}) \mid \mathcal{O}_{\mathrm{V}}[\![V]\!]\langle x\rangle) \\
&\sim \boldsymbol{\nu} q\,(\boldsymbol{\nu} x\,(\mathcal{O}[\![M']\!]\langle q\rangle \mid \mathcal{O}_{\mathrm{V}}[\![V]\!]\langle x\rangle) \\
&\qquad \mid q(y).\,\boldsymbol{\nu} w\,(\boldsymbol{\nu} x\,(\mathcal{O}_{\mathrm{V}}[\![V']\!]\langle w\rangle \mid \mathcal{O}_{\mathrm{V}}[\![V]\!]\langle x\rangle) \mid \mathsf{W}))) && \text{(C.34)}\\
&\succeq \boldsymbol{\nu} q\,(\mathcal{O}[\![M'\{x/V\}]\!]\langle q\rangle \mid q(y).\,\boldsymbol{\nu} w\,(\mathcal{O}_{\mathrm{V}}[\![V'\{x/V\}]\!]\langle w\rangle \mid \mathsf{W})) && \text{(C.35)}\\
&= \mathsf{rhs}
\end{aligned}$$

– $M = V'M'$. Then

$$\begin{aligned}
\mathsf{lhs} &= \boldsymbol{\nu} x\,(\boldsymbol{\nu} y\,(\mathcal{O}_{\mathrm{V}}[\![V']\!]\langle y\rangle \mid \boldsymbol{\nu} r\,(\mathcal{O}[\![M']\!]\langle r\rangle \mid r(w).\,\mathsf{W} \mid \mathcal{O}_{\mathrm{V}}[\![V]\!]\langle x\rangle) \\
&\sim \boldsymbol{\nu} y\,(\boldsymbol{\nu} x\,(\mathcal{O}_{\mathrm{V}}[\![V']\!]\langle y\rangle \mid \mathcal{O}_{\mathrm{V}}[\![V]\!]\langle x\rangle) \\
&\qquad \mid \boldsymbol{\nu} r\,(\boldsymbol{\nu} x\,(\mathcal{O}[\![M']\!]\langle r\rangle \mid \mathcal{O}_{\mathrm{V}}[\![V]\!]\langle x\rangle) \mid \mathsf{W}))) && \text{(C.36)}\\
&\succeq \boldsymbol{\nu} r\,(\mathcal{O}[\![M'\{x/V\}]\!]\langle r\rangle \mid \boldsymbol{\nu} r\,(\mathcal{O}_{\mathrm{V}}[\![V'\{x/V\}]\!]\langle r\rangle \mid \mathsf{W})) && \text{(C.37)}
\end{aligned}$$

– $M = x'V'$.

  * If $x' = x$, then $M\{x/V\} = V\ V'$. We have

$$\mathsf{lhs} = \boldsymbol{\nu} x\,(\overline{x}(z,q).\,(\mathcal{O}_{\mathrm{V}}[\![V']\!]\langle z\rangle \mid q \triangleright p) \mid \mathcal{O}_{\mathrm{V}}[\![V]\!]\langle x\rangle)$$

  We consider two cases.
  Suppose $V = z$, then $\mathcal{O}_{\mathrm{V}}[\![V]\!]\langle x\rangle = x \triangleright z$.
  Suppose $V = \lambda z.\,M'$, then $\mathcal{O}_{\mathrm{V}}[\![V]\!]\langle x\rangle = !x(z,q).\,\mathcal{O}[\![M']\!]\langle q\rangle$.

  * If $x' \neq x$, then $M\{x/V\} = x'\ V'$. We have

$$\begin{aligned}
\mathsf{lhs} &= \boldsymbol{\nu} x\,(\overline{x'}(z,q).\,(\mathcal{O}_{\mathrm{V}}[\![V']\!]\langle z\rangle \mid q \triangleright p) \mid \mathcal{O}_{\mathrm{V}}[\![V]\!]\langle x\rangle) \\
&\sim \overline{x'}(z,q).\,(\boldsymbol{\nu} x\,(\mathcal{O}_{\mathrm{V}}[\![V']\!]\langle z\rangle \mid \mathcal{O}_{\mathrm{V}}[\![V]\!]\langle x\rangle) \mid q \triangleright p) && \text{(C.38)}\\
&\succeq \overline{x'}(z,q).\,(\mathcal{O}_{\mathrm{V}}[\![V'\{x/V\}]\!]\langle z\rangle \mid q \triangleright p) = \mathsf{rhs} && \text{(C.39)}
\end{aligned}$$

– $M = (\lambda x'.\,M')V'$. Then

$$\begin{aligned}
\mathsf{lhs} &= \boldsymbol{\nu} x\,(\boldsymbol{\nu} y\,,w(\mathcal{O}_{\mathrm{V}}[\![\lambda x'.\,M']\!]\langle y\rangle \mid \mathcal{O}_{\mathrm{V}}[\![V']\!]\langle w\rangle \mid \mathsf{W}) \mid \mathcal{O}_{\mathrm{V}}[\![V]\!]\langle x\rangle) \\
&\sim \boldsymbol{\nu} y\,,w(\boldsymbol{\nu} x\,(\mathcal{O}_{\mathrm{V}}[\![\lambda x'.\,M']\!]\langle y\rangle \mid \mathcal{O}_{\mathrm{V}}[\![V]\!]\langle x\rangle) \\
&\qquad \mid \boldsymbol{\nu} x\,(\mathcal{O}_{\mathrm{V}}[\![V']\!]\langle w\rangle \mid \mathcal{O}_{\mathrm{V}}[\![V]\!]\langle x\rangle) \mid \mathsf{W}) && \text{(C.40)}\\
&\succeq \boldsymbol{\nu} y\,,w(\boldsymbol{\nu} x\,(\mathcal{O}_{\mathrm{V}}[\![(\lambda x'.\,M']\!]\langle y\rangle)\{x/V\} \mid \mathcal{O}_{\mathrm{V}}[\![V]\!]\langle x\rangle) \\
&\qquad \mid \boldsymbol{\nu} x\,(\mathcal{O}_{\mathrm{V}}[\![V'\{x/V\}]\!]\langle w\rangle \mid \mathcal{O}_{\mathrm{V}}[\![V]\!]\langle x\rangle) \mid \mathsf{W}) && \text{(C.41)}\\
&= \mathsf{rhs}
\end{aligned}$$

□

Lemma 25 (Validity of β_{v}-reduction)**.** *For any M, N in Λ, $M \longrightarrow N$ implies that for any p, $\mathcal{O}[\![M]\!]\langle p\rangle \succeq \mathcal{O}[\![N]\!]\langle p\rangle$.*

Proof. We show a stronger property, namely that $\mathcal{O}[\![M]\!]\langle p\rangle \xrightarrow{\tau}\succeq \mathcal{O}[\![N]\!]\langle p\rangle$. This is a consequence of Lemma 63, exploiting the congruence properties of $\succeq$, and the fact that for any M, C_{e} and p, the only transition $\mathcal{O}[\![C_{\mathrm{e}}[M]]\!]\langle p\rangle$ can do arises from a transition of the encoding of M (intuitively, the encoding of the hole is in active position). □

Lemma 26. *We have:*

$$\mathcal{O}[\![C_{\mathrm{e}}[xV]]\!]\langle p\rangle \sim \overline{x}(z,q).\,(\mathcal{O}_{\mathrm{V}}[\![V]\!]\langle z\rangle \mid q(y).\,\mathcal{O}[\![C_{\mathrm{e}}[y]]\!]\langle p\rangle).$$

Proof. We reason by induction over the shape of the evaluation context C_{e}.

- **Base case:** $C_\mathrm{e} = [\cdot]$. We observe $\mathcal{O}[\![y]\!]\langle p\rangle = \overline{p}(z).\, z \triangleright y$, hence $q(y).\, \mathcal{O}[\![y]\!]\langle p\rangle = q \triangleright p$.

  We then write

$$\begin{aligned}\mathcal{O}[\![xV]\!]\langle p\rangle &= \overline{x}(z,q).\,(\mathcal{O}_\mathrm{V}[\![V]\!]\langle z\rangle \mid q \triangleright p) \text{ by definition}\\ &= \overline{x}(z,q).\,(\mathcal{O}_\mathrm{V}[\![V]\!]\langle z\rangle \mid q(y).\,\mathcal{O}[\![y]\!]\langle p\rangle)\end{aligned}$$

- **Case** $C_\mathrm{e} = V'C'_\mathrm{e}$. We write

$$\begin{aligned}\mathcal{O}[\![C_\mathrm{e}[xV]]\!]\langle p\rangle &= \boldsymbol{\nu} s\,\big(\mathcal{O}_\mathrm{V}[\![V']\!]\langle s\rangle \mid \boldsymbol{\nu} r\,(\mathcal{O}[\![C'_\mathrm{e}[xV]]\!]\langle r\rangle \mid P_0)\big)\\ &\qquad \text{with } P_0 = r(w).\,\overline{s}(w',r').\,(w' \triangleright w \mid r' \triangleright p)\end{aligned}$$

  We have by induction

$$\mathcal{O}[\![C'_\mathrm{e}[xV]]\!]\langle r\rangle \sim \overline{x}(z_1,q_1).\,(\mathcal{O}_\mathrm{V}[\![V]\!]\langle z_1\rangle \mid q_1(y_1).\,\mathcal{O}[\![C'_\mathrm{e}[y_1]]\!]\langle r\rangle),$$

  which gives

$$\begin{aligned}\mathcal{O}[\![C_\mathrm{e}[xV]]\!]\langle p\rangle &\sim \boldsymbol{\nu} s\,\big(\mathcal{O}_\mathrm{V}[\![V']\!]\langle s\rangle \mid\\ &\qquad \boldsymbol{\nu} r\,(\overline{x}(z_1,q_1).\,(\mathcal{O}_\mathrm{V}[\![V]\!]\langle z_1\rangle \mid q_1(y_1).\,\mathcal{O}[\![C'_\mathrm{e}[y_1]]\!]\langle r\rangle) \mid P_0)\big)\\ &\sim \overline{x}(z_1,q_1).\,\boldsymbol{\nu} s\,\big(\mathcal{O}_\mathrm{V}[\![V']\!]\langle s\rangle\\ &\qquad \mid \boldsymbol{\nu} r\,(\mathcal{O}_\mathrm{V}[\![V]\!]\langle z_1\rangle \mid q_1(y_1).\,\mathcal{O}[\![C'_\mathrm{e}[y_1]]\!]\langle r\rangle \mid P_0)\big) &\text{(C.42)}\\ &\sim \overline{x}(z_1,q_1).\,\big(\mathcal{O}_\mathrm{V}[\![V]\!]\langle z_1\rangle\\ &\qquad \mid q_1(y_1).\,(\boldsymbol{\nu} s\,(\mathcal{O}_\mathrm{V}[\![V']\!]\langle s\rangle \mid \boldsymbol{\nu} r\,(\mathcal{O}[\![C'_\mathrm{e}[y_1]]\!]\langle r\rangle \mid P_0)))\big) &\text{(C.43)}\end{aligned}$$

  For (C.42), we observe that $\mathcal{O}_\mathrm{V}[\![V']\!]\langle s\rangle$ starts with an input at s, and P_0 starts with an input at r. Therefore, the bound output at x is the only possible transition for the process above, which allows us to bring the prefix on top.

  For (C.43), we recall that $P_0 = r(w).\,\overline{s}(w',r').\,(w' \triangleright w \mid r' \triangleright p)$. We observe that P_0 can start interacting only after the prefix $q_1(y_1)$ is triggered, because the only possible output at r is within $\mathcal{O}[\![C'_\mathrm{e}[y_1]]\!]\langle r\rangle$. In turn, because the output at s in P_0 is guarded by the input at r, the subterm $\mathcal{O}_\mathrm{V}[\![V']\!]\langle s\rangle$ can become active only after the interaction at r, and hence it is sound, modulo strong bisimilarity, to place $\mathcal{O}_\mathrm{V}[\![V']\!]\langle s\rangle$ under the prefix $q_1(y_1)$.

  We can then conclude, by observing that $\boldsymbol{\nu} s\,(\mathcal{O}_\mathrm{V}[\![V']\!]\langle s\rangle \mid \boldsymbol{\nu} r\,(\mathcal{O}[\![C'_\mathrm{e}[y_1]]\!]\langle r\rangle \mid P_0))$ is equal to $\mathcal{O}[\![V'C'_\mathrm{e}[y_1]]\!]\langle p\rangle$ by definition.

- **Case** $C_\mathrm{e} = C'_\mathrm{e}M$. We reason as follows:

$$\begin{aligned}\mathcal{O}[\![C_\mathrm{e}[xV]]\!]\langle p\rangle &= \boldsymbol{\nu} s\,(\mathcal{O}[\![C'_\mathrm{e}[xV]]\!]\langle s\rangle \mid s(z).\,\boldsymbol{\nu} r\,(\mathcal{O}[\![M]\!]\langle r\rangle \mid P_0))\\ &\qquad \text{with } P_0 = r(w).\,\overline{z}(w',r').\,(w' \triangleright w \mid r' \triangleright p)\\ &\sim \boldsymbol{\nu} s\,\big(\overline{x}(z_1,q_1).\,(\mathcal{O}_\mathrm{V}[\![V]\!]\langle z_1\rangle \mid q_1(y_1).\,\mathcal{O}[\![C'_\mathrm{e}[y_1]]\!]\langle s\rangle)\\ &\qquad \mid s(z).\,\boldsymbol{\nu} r\,(\mathcal{O}[\![M]\!]\langle r\rangle \mid P_0)\big) &\text{(C.44)}\\ &\sim \overline{x}(z_1,q_1).\,\big(\mathcal{O}_\mathrm{V}[\![V]\!]\langle z_1\rangle \mid q_1(y_1).\,\boldsymbol{\nu} s\,(\mathcal{O}[\![C'_\mathrm{e}[y_1]]\!]\langle s\rangle)\\ &\qquad \mid s(z).\,\boldsymbol{\nu} r\,(\mathcal{O}[\![M]\!]\langle r\rangle \mid P_0))\big) &\text{(C.45)}\end{aligned}$$

  Step (C.44) follows by induction, and step (C.45) is deduced as in the previous case.

□

Appendix C.3. Completeness

Lemma 34. *If* C_e *is an evaluation context,* V *is a value,* x *is a name and* z *is fresh in* C_e*, then*

$$\mathcal{I}[\![C_\mathrm{e}[xV]]\!] \approx \mathcal{I}[\![(\lambda z.\, C_\mathrm{e}[z])(xV)]\!].$$

Proof. We use Lemma 26 and 24; we get:

$$\mathcal{I}[\![C_\mathrm{e}[xV]]\!]\langle p\rangle \approx \overline{x}(z,q).\,(\mathcal{O}_\mathrm{V}[\![V]\!]\langle z\rangle \mid q(y).\,\mathcal{I}[\![C_\mathrm{e}[y]]\!]\langle p\rangle)$$

and

$$\mathcal{I}[\![(\lambda w.\, C_\mathrm{e}[w])(xV)]\!]\langle p\rangle \approx \overline{x}(z,q).\,(\mathcal{O}_\mathrm{V}[\![V]\!]\langle z\rangle \mid q(y).\,\mathcal{I}[\![(\lambda w.\, C_\mathrm{e}[w])y]\!]\langle p\rangle)$$

We conclude by validity of β-reduction (Lemma 36) applied to $\mathcal{I}[\![(\lambda w.\, C_\mathrm{e}[w])y]\!]\langle p\rangle$. □

$$
\begin{array}{ll}
M \Uparrow \text{ and } N \Uparrow: & X_{M,N} = (\widetilde{y})\ \mathcal{V}[\![\Omega]\!] \\
M \Downarrow x \text{ and } N \Downarrow x: & X_{M,N} = (\widetilde{y})\ \mathcal{V}[\![x]\!] \\
M \Downarrow \lambda x.\, M' \text{ and } N \Downarrow \lambda x.\, N': & X_{M,N} = (\widetilde{y})\ \mathcal{V}[\![\lambda x.\, X_{M',N'}]\!] \\
M \Downarrow C_{\mathrm{e}}[xV] \text{ and } N \Downarrow C'_{\mathrm{e}}[xV']: & X_{M,N} = (\widetilde{y})\ \mathcal{V}[\![(\lambda z.\, X_{C_{\mathrm{e}}[z],C'_{\mathrm{e}}[z]})\ (x\ X_{V,V'})]\!] \\
M \Downarrow x,\ N \Downarrow \lambda z.\, N',\ N' \Downarrow C_{\mathrm{e}}[xV]: & X_{M,N} = (\widetilde{y})\ \mathcal{V}[\![\lambda z.\, ((\lambda w.\, X_{w,C_{\mathrm{e}}[w]})\ (x\ X_{z,V}))]\!] \\
M \Downarrow \lambda z.\, M',\ M' \Downarrow C_{\mathrm{e}}[xV],\ N \Downarrow x: & X_{M,N} = (\widetilde{y})\ \mathcal{V}[\![\lambda z.\, ((\lambda w.\, X_{C_{\mathrm{e}}[w],w})\ (x\ X_{V,z}))]\!]
\end{array}
$$

Figure D.7: System $\mathcal{E}^{\mathbf{L}}_{\mathcal{R}}$ of equations (the last two equations are only needed for $\leftrightarroweq_\eta$)

$$
\begin{array}{ll}
M \Uparrow \text{ and } N \Uparrow: & X_{M,N} = (\widetilde{y}, p)\ \mathbf{0} \\
M \Downarrow C_{\mathrm{e}}[xv] \text{ and } N \Downarrow C'_{\mathrm{e}}[xv']: & X_{M,N} = (\widetilde{y}, p)\ (\boldsymbol{\nu} z, q\ )(\overline{x}\langle z, q\rangle \mid X^{\mathcal{V}}_{V,V'}\langle z, \widetilde{y}'\rangle \\
& \qquad \mid q(w).\, X_{C_{\mathrm{e}}[w],C'_{\mathrm{e}}[w]}\langle \widetilde{y}'', p\rangle) \\
M \Downarrow V \text{ and } N \Downarrow V': & X_{M,N} = (\widetilde{y}, p)\ (\boldsymbol{\nu} y\ )(\overline{p}\langle y\rangle \mid X^{\mathcal{V}}_{v,v'}\langle z, \widetilde{y}'\rangle) \\
V = x \text{ and } V' = x: & X^{\mathcal{V}}_{x,x} = (z, x)\ z \blacktriangleright x \\
V = \lambda x.\, M \text{ and } V' = \lambda x.\, N: & X^{\mathcal{V}}_{\lambda x.M,\lambda x.N} = (z, \widetilde{y})\ !z(x, q).\, X_{M,N}\langle \widetilde{y}', q\rangle \\
V = x,\ V' = \lambda z.\, N,\ N \Downarrow C_{\mathrm{e}}[xV]: & X^{\mathcal{V}}_{x,\lambda z.N} = (y_0, \widetilde{y})\ !y_0(z, q).\, (\boldsymbol{\nu} z', q'\ ) \\
& \qquad (\overline{x}\langle z', q'\rangle \mid X^{\mathcal{V}}_{z,V}\langle z', \widetilde{y}'\rangle \\
& \qquad\quad \mid q'(w).\, X_{w,C_{\mathrm{e}}[w]}\langle \widetilde{y}'', q\rangle) \\
V = \lambda z.\, M,\ M \Downarrow C_{\mathrm{e}}[xV],\ V' = x: & X^{\mathcal{V}}_{\lambda z.M,x} = (y_0, \widetilde{y})\ !y_0(z, q).\, (\boldsymbol{\nu} z', q'\ ) \\
& \qquad (\overline{x}\langle z', q'\rangle \mid X^{\mathcal{V}}_{V,z}\langle z', \widetilde{y}'\rangle \\
& \qquad\quad \mid q'(w).\, X_{C_{\mathrm{e}}[w],w}\langle \widetilde{y}'', q\rangle)
\end{array}
$$

Figure D.8: System $\mathcal{E}^{\mathbf{L}'}_{\mathcal{R}}$ of equations (the last two equations are only needed for $\leftrightarroweq_\eta$)

Appendix D. Systems of equations for AL$\boldsymbol{\pi}$ (Section 5)

The systems of equations for ALπ are presented on Figures D.7 and D.8.

To introduce the second system of equations, we define the extension of the encoding to equation variables as follows:

$$
\mathcal{V}[\![X_{M,N}]\!] \stackrel{\text{def}}{=} (p)\ X_{M,N}\langle \widetilde{y}, p\rangle \quad \text{where } \widetilde{y} = \mathsf{fv}(M, N)
$$

Appendix E. Unique solution techniques for contextual relations (Section 6)

The proof of the following lemma is very similar to the proof of Theorem 16. For more details, we refer the reader to [7], particularly the proof of unique solution for weak bisimilarity in the setting of CCS.

Lemma 55 (Post-fixed points, $\preceq_{\mathrm{tr}}$)**.** *Let $\mathcal{E}$ be a guarded system of equations, and $\widetilde{\mathrm{K}}_{\mathcal{E}}$ its syntactic solution. Suppose $\widetilde{\mathrm{K}}_{\mathcal{E}}$ has no divergences. If $\widetilde{F}$ is a post-fixed point for $\preceq_{\mathrm{tr}}$ of $\mathcal{E}$, then $\widetilde{F} \preceq_{\mathrm{tr}} \widetilde{\mathrm{K}}_{\mathcal{E}}$.*

Proof. For simplicity, we only give the proof for a single equation E, rather than a system of equations. Generalisation to systems of equations does not add any particular difficulty.

Assume E is an equation, F an abstraction, and $F \preceq_{\mathrm{tr}} E[P]$. We fix a set of fresh names $\widetilde{a}$, and write P for $F\langle\widetilde{a}\rangle$. If $\widetilde{\alpha} = \alpha_1 \ldots \alpha_n$ is a finite trace of P, we build a growing sequence of transitions of $E^n\langle\widetilde{a}\rangle$ such that $E^n[F]\langle\widetilde{a}\rangle \xRightarrow{\alpha_1\ldots\alpha_{i_k}} E_n[F] \xRightarrow{\alpha_{i_k+1},\ldots,\alpha_n} P_n$.

We start by making two observations:

1. If the transitions in $\widetilde{\alpha}$ are all transitions of the context $E^n\langle\widetilde{a}\rangle$, we stop and we have $E^n\langle\widetilde{a}\rangle \stackrel{\widetilde{\alpha}}{\Longrightarrow}$, and thus $K_E\langle\widetilde{a}\rangle \stackrel{\widetilde{\alpha}}{\Longrightarrow}$. So $\widetilde{\alpha}$ is a trace of $K_E\langle\widetilde{a}\rangle$.

2. Otherwise there is an infinite sequence of transitions from $K_E\langle\widetilde{a}\rangle$ with visible actions $\alpha_1, \ldots, \alpha_{i_k}$ for some k; therefore $K_E\langle\widetilde{a}\rangle$ has a divergence.

We now explain the construction of the sequence. Assume for that that we have both $(i):\ E^n\langle\widetilde{a}\rangle \xRightarrow{\alpha_1,\ldots,\alpha_{i_k}} E_n$ and $(ii):\ E_n[F] \xRightarrow{\alpha_{i_k+1},\ldots,\alpha_n}$.

By (i) it follows that $E^{n+1}[F]\langle\widetilde{a}\rangle \xRightarrow{\alpha_1,\ldots,\alpha_{i_k}} E_n[E[F]]$.

By (ii) and congruence of $\preceq_{\rm tr}$, it follows that $\alpha_{i_k+1}, \ldots, \alpha_n$ is a trace of $E_n\langle\widetilde{a}\rangle[F] \preceq_{\rm tr} E_n[E[F]]$.

We take for the new sequence of transitions the concatenation of the previous one, and the part of $E_n[E[F]] \xRightarrow{\alpha_{i_k+1},\ldots,\alpha_n}$ that is a transition of the context $E_n[E]$. Since E is weakly guarded, this is not an empty sequence.

By observation 2 above this construction has to stop, otherwise there would be a divergence. We conclude by observation 1.

□