What's TyCO, After All?

Maxime Gamboni

April 20, 2009

Abstract

We study the expressive power of asynchronous π-calculus with nested variants (π_{a}^{V}), of uniform and non-uniform TyCO by means of encodings.

I was given a draft of this report and had to make the necessary changes to the theory in order to make the theorems true, and prove them

Uniform TyCO can be seen as a sub-calculus of π_{a}^{V}, the only difference being that π_{a}^{V} requires a separate construct for analysing an input while in TyCO input and value analysis are tightly bound. Still, we can give an encoding (embedding) that is fully abstract and good.

We propose an encoding from local π_{a}^{V} to TyCO and show that it is fully abstract.

1 Introduction

A Question of Expressive Power TyCO is an asynchronous name-passing process calculus for typed concurrent objects, proposed by Vasconcelos [Vas94]. In comparison to a standard π-calculus [MPW92], TyCO-processes are indeed more like objects in the sense that *every* communication exchanges a structured value, reminiscent of method invocation, consisting of an *name* tagged with a *label.* More recently, Sangiorgi conceived a π-calculus equipped with *labeled values*, called variants, which has been brought up as a suitable vehicle for the description, by means of encodings, of the semantics of typed higher-level object calculi [San98]. Apparently, TyCO is just an asynchronous version of the π-calculus with variants, which we refer to as π_{a}^{V}. Indeed, as we will show in this paper, TyCO represents a proper sub-calculus of π_{a}^{V} in a precise sense. The obvious question arises, whether this sub-calculus is less expressive.

My Diploma Project

I was given a draft of this report, containing chapters one to five, ie

- syntax and semantics of the calculi
- receptiveness theory applied to TyCO
- an encoding of π_{a}^{V} to TyCO

I had to make necessary changes to the theory to make the theorems true and write their proofs, in order to have a fully abstract encoding of π_{a}^{V} to TyCO with the proof of this fact.

Here is a list of most important changes and additions that I did :

1. *Case Reduction Relation*
   We tried three different semantics for handling case reduction (see Definition 2.2.1).
   The original semantics was handling it as part of Structural Congruence (Definition 2.1.1) but it was breaking subject congruence (2.1.3).
   We tried to put it as a τ-transition but the full abstraction of TyCO-π_{a}^{V} embedding (Proposition 3.0.3) worked for weak equivalences only.
   So we created this directional congruence $\rightarrowtail$ (Definition 2.2.1)
2. *Linear Weakening*
   In the receptiveness theory applied to TyCO (section 3), we had weakening on uniform names only, but the type soundness (Proposition 4.1.6) was broken in the following case :
   Let $\Theta \vdash_{\mathrm{R}} P = (\boldsymbol{\nu} a)\,(a!\mathrm{l_k}(\boldsymbol{\nu} b).A \mid a?\{\mathrm{l}_j(x_j){=}P_j \;\mid\; j{\in}J\})$ with a linear.
   $P \xrightarrow{\tau} (\boldsymbol{\nu} a)\,(\boldsymbol{\nu} b)\,(A \mid P_j\{^{b}/_{x_j}\}) = P'$.
   (R-Res) of Table 5 requires a to be in Γ_1 and Δ_1 for the subprocess restricted by $(\boldsymbol{\nu} a)$, where a has been consumed and is no longer free.
   So without linear weakening there is no Θ' with $\Theta' \vdash_{\mathrm{R}} P'$.
3. *Undecidability of* (R-Link)
   We had introduced the concept of *Dynamic Links* to avoid extrusion of plain names (Section 4.1)
   I spent a few weeks to prove its (receptive) typability (Table 7) before seeing that it is undecidable (so I made it an axiom) (A partial proof is given in the appendix B.6)

4. *Definition of Receptive Equivalences*
   I adapted the definition of weak receptive bisimulation (4.2.6) to strong receptive bisimulation and expansion (4.2.7 and 4.2.8)
5. *Made the Nested Encoding Syntax-Directed*
   The encoding from π_{a}^{V} to TyCO was type-directed but that was not necessary so it could be simplified.
6. *Proofs*
   I wrote the appendix B containing the proof of most lemmas, propositions and theorem.
7. *and more*
   The above changes required updating some propositions

There is a rule missing in the operational semantics (Table 2) for free communication:

$$(\textsc{FCom}_1)\ \frac{P_1 \xrightarrow{c!v} P_1' \qquad P_2 \xrightarrow{c?v} P_2'}{P_1|P_2 \xrightarrow{\tau} (P_1'|P_2')}$$

The absence of this rule breaks the full-abstraction of the π_{a}^{V}-TyCO for some kind of processes, as explained in Proposition 5.2.6 and Theorem 5.2.7.

It seems however that adding this rule would keep all propositions valid and make the side condition of π_{a}^{V}-TyCO operational correspondence (Proposition 5.2.6) unnecessary.

Finally, I did not have time to fix the non-local encoding that is mentioned in the last section.

The Contribution of this Paper

We set out to formally underline the similarity—and difference—between the two calculi. We do so by supplying two mutual encodings and studying their properties. First, the usual syntax presentation of the two calculi is changed such that they become very much alike. Then, we present a straightforward encoding from TyCO into π_{a}^{V} that makes explicit that (i) the former is a subcalculus of the latter, in that TyCO only knows values with precisely one level of label nesting; (ii) the difference in the atomicity of communication bears no problems, at least in the standard uniform setting. As a side-effect, the encoding allows to import to TyCO the theory of π_{a}^{V}.

One Answer We first present an encoding of a *Local* π_{a}^{V} into TyCO, where local means that outputs are bound and a process can't receive on a received name. This shows that nested variants can be encoded in one-level-only variants . We prove that this encoding is indeed fully abstract with respect to $\approx$ in the source and $\approx_{\mathrm{R}}$ in the target, which is defined on an adaptation to our asynchronous setting of Sangiorgi's theory of receptiveness [San99a, San99b]. So, yes, although TyCO is a proper sub-calculus of π_{a}^{V}, no expressive power is lost in the above formal sense.

We then give hints on the way the local encoding can be modified to work with non-local terms as well.

2 Preliminaries

2.1 Syntax

Terms Let $a, b, c, x \in \mathbf{N}$ be *names.* Let $\mathrm{l} \in \mathbf{L}$ be a *label.* Let J range over finite indexing sets. We assume the convention that names are used as channels a or variables x. A *value* v is a name optionally tagged with one or more labels. Then, *processes* P for both π_{a}^{V} and TyCO are given by the grammar in Table 1 for a common part such that their difference only arises for the definition of *normal* processes: in essence, output and input in TyCO always mention simply labeled values, while in π_{a}^{V} the exchanged values could be just names or arbitrarily nested variant values. This, of course, is also represented in the admitted type expressions. Finally, TyCO input is destructive. while π_{a}^{V} needs a case operator as explicit destructor for variant values.

We make the standard assumptions on operator precedence and use the standard definition of free names $\mathrm{fn}(P)$ and bound names $\mathrm{bn}(P)$ of a process P, considering the operators for restriction, input, replication, and case analysis, as binders for the names x, x_j.

Definition 2.1.1. Structural congruence*, written $\equiv$, is the smallest relation congruence satisfying the axioms below:*

- $P \equiv Q$*, if* P *is* α*-convertible to* Q*;*
- $P|\mathbf{0} \equiv P$, $P|Q \equiv Q|P$, $(P|Q)|R \equiv P|(Q|R)$*;*
- $(\boldsymbol{\nu} a)\, \mathbf{0} \equiv \mathbf{0}$, $(\boldsymbol{\nu} a)\, (\boldsymbol{\nu} b)\, P \equiv (\boldsymbol{\nu} b)\, (\boldsymbol{\nu} a)\, P$*,*
- $(\boldsymbol{\nu} a)\, (P|Q) \equiv P|(\boldsymbol{\nu} a)\, Q$*, if* $a \notin \mathrm{fn}(P)$*;*

The difference in atomicity of communication shows up already in the definition of *errors* in the two calculi, both of which are based on non-matching

Common Part

P	$::=$	$\mathbf{0}$	inactive process
	$\mid$	N	normal process
	$\mid$	$P\|P$	parallel composition
	$\mid$	$(\boldsymbol{\nu} x)\, P$	restriction

π_{a}^{V}

N	$::=$	$a!v$	output
	$\mid$	$a?(x).P$	input
	$\mid$	$a?^*(x).P$	replicated input
	$\mid$	$\mathsf{case}\ v\ \mathsf{of}\ \{\mathrm{l}_j(x_j){=}P_j \ \mid\ j{\in}J\}$	variant destructor
v	$::=$	a	name
	$\mid$	$\mathrm{l}\langle v\rangle$	labeled value
T	$::=$	X	type variable
	$\mid$	$[T]$	channel type
	$\mid$	$\{\mathrm{l}_j{:}T_j \ \mid\ j{\in}J\}$	variant type
	$\mid$	$\mu X.T$	recursive type

TyCO

N	$::=$	$a!\mathrm{l}\langle b\rangle$	“invocation”
	$\mid$	$a?\{\mathrm{l}_j(x_j){=}P_j \ \mid\ j{\in}J\}$	“object”
	$\mid$	$a?^*\{\mathrm{l}_j(x_j){=}P_j \ \mid\ j{\in}J\}$	“replicated object”
T	$::=$	X	type variable
	$\mid$	$[\{\mathrm{l}_j{:}T_j \ \mid\ j{\in}J\}]$	object type
	$\mid$	$\mu X.T$	recursive type

Table 1: Syntax

labels: while π_{a}^{V} checks for the availability of a label only when performing an explicit case analysis on a possibly formerly received labeled value, TyCO checks for the availability of a (method) label already when trying to transmit a labeled value. In π_{a}^{V} and uniform TyCO, this subtle difference does not play an important role, but non-uniformity exploits it.

Definition 2.1.2 (Errors)**.**

1. $\mathrm{Error}_{\pi} \overset{\mathrm{def}}{=} \{\, P \in \pi_{\mathrm{a}}^{V} \mid \ P \equiv (\boldsymbol{\nu}\tilde{x})\,(\mathsf{case}\,\mathrm{l}\langle v\rangle\,\mathsf{of}\,\{\mathrm{l}_j(x_j){=}P_j \ \mid\ j{\in}J\} \mid Q)$ *and* $\mathrm{l} \neq \mathrm{l}_j$ *for all* $j{\in}J \,\}$
2. $\mathrm{Error}_{T} \overset{\mathrm{def}}{=} \{\, P \in \mathit{TyCO} \mid \big(P \equiv (\boldsymbol{\nu}\tilde{x})\,(a!\mathrm{l}\langle b\rangle \mid a?\{\mathrm{l}_j(x_j){=}P_j \ \mid\ j{\in}J\} \mid Q)$ *or* $P \equiv (\boldsymbol{\nu}\tilde{x})\,(a!\mathrm{l}\langle b\rangle \mid a?^{*}\{\mathrm{l}_j(x_j){=}P_j \ \mid\ j{\in}J\} \mid Q)\big)$ *and* $\mathrm{l} \neq \mathrm{l}_j$ *for all* $j{\in}J \,\}$

Note that the notions of error need to be refined in a polyadic setting to accommodate for arity mismatches.

Types In π_{a}^{V}, we have free nesting of type constructors, which means that

$$\begin{aligned} T_1 &\overset{\mathrm{def}}{=} [[T]] \\ T_2 &\overset{\mathrm{def}}{=} \{\mathrm{l}_1{:}\{\mathrm{k}_1{:}T_1, \mathrm{k}_2{:}T_2\}, \mathrm{l}_2{:}T_3\} \end{aligned}$$

are well-formed types. In contrast, types in TyCO are such that there is strict alternation between channel and variant types, as expressed in the shape of object types:

- each channel carries a branching structure, and
- each type inside a label is a channel type (or a base type).

This notation also indicates the tightly joint occurrence of communication and selection in TyCO. In particular, the π_{a}^{V}-types T_1 and T_2 mentioned above can not be formed within TyCO.

Type equality $T_1{=}T_2$, in the context of recursive types, holds if T_1 and T_2 denote the same infinite tree unfolding. Trees have nodes of type channel $[\cdots]$ with exactly one outgoing edge (monadic) and nodes of type variant $\{\cdots\}$ with labeled outgoing edges for each variant tag. The leaves are empty variants, i.e., without outgoing edges.

Simple Type System Judgments are of the form $\Gamma \vdash_S P$, where Γ is a finite partial function from names to types covering all the free names of P. Judgments are generated by the rules in Table 10 of Appendix A for TyCO and π_a^V in a standard and completely analogous fashion. A TyCO (or π_a^V) process P is called *simply well-typed* if there is a typing context Γ such that $\Gamma \vdash_S P$ can be derived from the typing rules of TyCO (or π_a^V) in Table 10.

Typing is preserved under structural congruence.

Lemma 2.1.3 (Subject Congruence). *If* $\Sigma \vdash_S P \equiv Q$*, then* $\Sigma \vdash_S Q$.

Typing guarantees the absence of current errors.

Proposition 2.1.4 (Absence of Errors). *If* $\Sigma \vdash_S P$*, then* $P \not\in \mathrm{Error}$.

Originally, π_a^V was introduced with a subtyping system based on the directions in which channel names can be used by processes.

In this paper, for simplicity, we omit subtyping completely; we strongly conjecture that our formal comparison of the two calculi would also be valid in the context of subtyping.

In the remainder of the paper, we only consider simply well-typed terms. Moreover, whenever we consider a well-typed term in π_a^V, we follow the convention to use the name u to denote *variant variables*, i.e., names of type variant that occur in binding position of input, replication, or case constructs.

2.2 Operational Semantics

We need a relation for processing case reductions in π_a^V.

Definition 2.2.1. Case Reduction*, written* $\rightarrowtail$*, is defined like this :*

- $\mathsf{case}\, \mathrm{l}_k\langle v\rangle\ \mathsf{of}\ \{\mathrm{l}_j(x_j){=}P_j \mid j{\in}J\} \rightarrowtail P_k\{^v/_{x_k}\}$*, if* $k{\in}J$.
- *if* $P \rightarrowtail P'$ *then (for any process* Q *and name* a*)*
    - $P \mid Q \rightarrowtail P' \mid Q$,
    - $Q \mid P \rightarrowtail Q \mid P'$ *and*
    - $(\boldsymbol{\nu} a)\, P \rightarrowtail (\boldsymbol{\nu} a)\, P'$.

We give the operational semantics as a labeled transition system generated by the rules in Table 2, which uses many rules commonly for TyCO and π_a^V, assuming that labels are of the form:

$$\mu \ ::= \ \tau \ \mid \ c?v \ \mid \ (\boldsymbol{\nu} b)\, c!v \ \mid \ c!v$$

Common Part

$$(\textsc{Out})\ \frac{-}{c!v \xrightarrow{c!v} \mathbf{0}}$$

$$(\textsc{Open})\ \frac{P \xrightarrow{c!v} P'}{(\boldsymbol{\nu} b)\, P \xrightarrow{(\boldsymbol{\nu} b)\, c!v} P'}\ b \in \mathrm{n}(v)\backslash\{c\}$$

$$(\textsc{Com}_1)\ \frac{P_1 \xrightarrow{(\boldsymbol{\nu} b)\, c!v} P_1' \qquad P_2 \xrightarrow{c?v} P_2'}{P_1|P_2 \xrightarrow{\tau} (\boldsymbol{\nu} b)\, (P_1'|P_2')}\ b \notin \mathrm{fn}(P_2)$$

$$(\textsc{Par}_1)\ \frac{P_1 \xrightarrow{\mu} P_1'}{P_1 \mid P_2 \xrightarrow{\mu} P_1' \mid P_2}\ \mathrm{bn}(\mu) \cap \mathrm{fn}(P_2) = \emptyset$$

$$(\textsc{Res})\ \frac{P \xrightarrow{\mu} P'}{(\boldsymbol{\nu} a)\, P \xrightarrow{\mu} (\boldsymbol{\nu} a)\, P'}\ a \notin \mathrm{n}(\mu)$$

$$(\textsc{Alpha})\ \frac{P \text{ alpha-convertible to } P' \qquad P' \xrightarrow{\mu} P''}{P \xrightarrow{\mu} P''}$$

π_{a}^{V}

$$(\textsc{Inp})\quad \frac{-}{c?(x).P \xrightarrow{c?v} P\{{}^{v}\!/\!_{x}\}}$$

$$(\textsc{Rep})\quad \frac{-}{c?^{*}(x).P \xrightarrow{c?v} P\{{}^{v}\!/\!_{x}\} \mid c?^{*}(x).P}$$

$$(\text{P-}\textsc{Case})\quad \frac{P_k\{{}^{v}\!/\!_{x_k}\} \xrightarrow{\mu} Q \qquad k \in J}{\mathsf{case}\ \mathrm{l}_k\langle v\rangle\ \mathsf{of}\ \{\mathrm{l}_j(x_j){=}P_j \ \mid\ j{\in}J\} \xrightarrow{\mu} Q}$$

TyCO

$$(\textsc{Inp})\quad \frac{k \in J}{c?\{\mathrm{l}_j(x_j){=}P_j \ \mid\ j{\in}J\} \xrightarrow{c?\mathrm{l}_k\langle v\rangle} P_k\{{}^{v}\!/\!_{x_k}\}}$$

$$(\textsc{Rep})\quad \frac{k \in J}{c?^{*}\{\mathrm{l}_j(x_j){=}P_j \ \mid\ j{\in}J\} \xrightarrow{c?\mathrm{l}_k\langle v\rangle} P_k\{{}^{v}\!/\!_{x_k}\} \mid c?^{*}\{\mathrm{l}_j(x_j){=}P_j \ \mid\ j{\in}J\}}$$

Table 2: Labeled Transition System

where TyCO-transition labels arise as special cases for $v = \mathrm{l}_k\langle b\rangle$. (Note that this implies that there will never be more than one name restricted on a transition label.) As expected, we only need separate rules for dealing with input, replication, and case analysis, in the respective calculi.

The name in a value (there is always exactly one) is written $\mathrm{n}(v)$.

In $(\boldsymbol{\nu} b)\, c!v$ we require that $c \notin \mathrm{n}(v)$.

The definitions of $\mathrm{fn}(\mu)$, $\mathrm{bn}(\mu)$ and $\mathrm{n}(\mu)$ are as follows :

- $\mathrm{fn}(\tau) = \mathrm{bn}(\tau) = \emptyset$,
- $\mathrm{fn}(c?v) = \{c\}$, $\mathrm{bn}(c?v) = \mathrm{n}(v)$,
- $\mathrm{fn}((\boldsymbol{\nu} b)\, c!v) = \{c\}$, $\mathrm{bn}((\boldsymbol{\nu} b)\, c!v) = \{b\}$,
- $\mathrm{fn}(c!v) = \{c\} \cup \mathrm{n}(v)$, $\mathrm{bn}(c!v) = \emptyset$,
- $\mathrm{n}(\mu) = \mathrm{fn}(\mu) \cup \mathrm{bn}(\mu)$.

Note that in our semantics case analysis does not take a step, as opposed to [San98], which simplifies our technical work later on.

Substitution $P\{{}^{a}\!/\!{}_{b}\}$ is defined in the standard capture avoiding way. As usual, τ-transitions are often referred to as *reductions* while omitting the τ. The relation $\Rightarrow$ is the reflexive-transitive closure of $\longrightarrow$, while $\stackrel{\mu}{\Longrightarrow}$ is $\Rightarrow \xrightarrow{\mu} \Rightarrow$.

The merits of the simple type systems in both TyCO and π_{a}^{V} are that well-typedness is preserved under reduction, and thus that well-typed processes can never lead to an error state.

Proposition 2.2.2. *Let* $\Sigma \vdash_{\mathrm{S}} P$.
If $P \rightarrowtail P'$*, then there is* Σ' *with* $\Sigma' \vdash_{\mathrm{S}} P'$.

Proposition 2.2.3 (Subject Reduction)**.** *Let* $\Sigma \vdash_{\mathrm{S}} P$*. If* $P \longrightarrow P'$*, then there is* Σ' *with* $\Sigma' \vdash_{\mathrm{S}} P'$.

A more detailed version is found in appendix : See Proposition A.0.3

Corollary 2.2.4 (Type Safety)**.** *If* $\Sigma \vdash_{\mathrm{S}} P \Rightarrow P'$*, then* $P' \notin \mathrm{Error}$.

A TyCO process is typable with our type system if, and only if, it is typable with Vasconcelos' type system [Vas94].

The following Lemma tells that case reduction is somehow "transparent" :

Lemma 2.2.5. *1. If* $P \rightarrowtail P_0 \xrightarrow{\mu} P_0'$ *then either* $P \xrightarrow{\mu} P_0'$ *or* $\exists P'$ *s.t.* $P \xrightarrow{\mu} P' \rightarrowtail P_0'$*)*

2. *If* $P \xrightarrow{\mu} P'$ *and* $P \rightarrowtail P_0$ *then either* $P_0 \xrightarrow{\mu} P'$ *or* $\exists P_0'$ *s.t.* $P_0 \xrightarrow{\mu} P_0'$ *and* $P' \rightarrowtail P_0'$.

Proof. In Appendix : B.2

□

Corollary 2.2.6. *If* $P \rightarrowtail P'$ *then* $P \sim P'$.

2.3 Behavioral Semantics

We will use the concept of *process contexts* of which the definition is standard :

Definition 2.3.1 (Process Contexts). *A Process Context* $C[\cdot]$ *is a process which contains exactly one instance of a* hole $[\cdot]$, *i.e. the syntax is as follows :*

$$C \ ::= \ [\cdot] \ \ | \ \ C|P \ \ | \ \ P|C \ \ | \ \ (\boldsymbol{\nu} x)\,C \ \ | \ \ \ldots$$

(input and case expression omitted, they similarly take care that exactly one hole is present in the process syntax)

The application $C[P]$ *of a process* P *to a context* $C[\cdot]$ *is defined by replacing* $[\cdot]$ *by* P *in the definition of* $C[\cdot]$.

We will use the standard (synchronous) definitions of strong ($\sim$) and weak ($\approx$) bisimulation, barbed bisimulation ($\dot{\approx}$) and barbed congruence ($\cong$).

They can for instance be found in [Bor98].

2.4 Properties of Encodings

In this paper, we present several *encodings*, i.e., inductively defined mappings from the syntax of some source calculus into the syntax of some target calculus. Apart from mere compositionality, there are several other requirements to the usefulness of an encoding, which we here recall rather informally:

- An encoding is called *fully abstract* if it preserves (*full*) and reflects (*abstract*) standard equivalences.
- An encoding is *good* if it is name-preserving and distributed (like in Table 3, parallel composition is translated into the mere parallel composition of the translated components), as well as deadlock- and divergence-free, which refers to criteria introduced by Palamidessi to the domain of encodings between π-calculi [Pal97].

$[\![\mathbf{0}]\!]$	$\stackrel{\text{def}}{=}$	$\mathbf{0}$
$[\![P_1 \vert P_2]\!]$	$\stackrel{\text{def}}{=}$	$[\![P_1]\!] \mid [\![P_2]\!]$
$[\![(\boldsymbol{\nu} x)\, P]\!]$	$\stackrel{\text{def}}{=}$	$(\boldsymbol{\nu} x)\, [\![P]\!]$

Table 3: Distributed Encodings

$[\![a!\mathrm{l}\langle b\rangle]\!]$	$\stackrel{\text{def}}{=}$	$a!\mathrm{l}\langle b\rangle$
$[\![a?\{\mathrm{l}_j(x_j){=}P_j \mid j{\in}J\}]\!]$	$\stackrel{\text{def}}{=}$	$a?(u).\mathsf{case}\, u\, \mathsf{of}\, \{\mathrm{l}_j(x){=}[\![P_j]\!] \mid j{\in}J\}$
$[\![a?^*\{\mathrm{l}_j(x_j){=}P_j \mid j{\in}J\}]\!]$	$\stackrel{\text{def}}{=}$	$a?^*(u).\mathsf{case}\, u\, \mathsf{of}\, \{\mathrm{l}_j(x){=}[\![P_j]\!] \mid j{\in}J\}$
$[\![T]\!]$	$\stackrel{\text{def}}{=}$	T

Table 4: Encoding of TyCO into π_{a}^{V}

While the former criterion is interesting for more theoretical purposes and to prove properties of source terms by means of translation into an "underlying" calculus, the latter indicates the actual implementability of the source within the target. Often, one can only achieve fully abstract encodings at the expense of goodness (c.f. [NP00, Nes00]).

In this paper, the source and target calculi are quite similar: all our encodings build on a common core, as shown in Table 3.

3 Embedding TyCO into π_{a}^{V}

Uniform TyCO resembles just a particular sub-calculus of nested π_{a}^{V}. In addition to the core mapping in Table 3, the encoding clauses of Table 4, where the variable u is assumed to be fresh, make explicit the fact that communication of a labeled value and selection of a continuation based on a case analysis, which are performed atomically in TyCO, have to be modeled as separate subsequent operations in π_{a}^{V}, which naturally appears solely on the receiver side of communication. The trivial translation of types underlines the fact that the encoding represents an embedding of the source language TyCO into the target language π_{a}^{V}.

This encoding is both *good* and *fully abstract*, with respect to any kind of equivalence and type-respecting contexts. Note that the encoding trivially preserves types and well-typedness of terms.

The formal statements build on the exhibition of an *operational correspondence* result between the transitions of terms and the transitions of their translations.

Proposition 3.0.1. *Let P be a TyCO term.*

- *If $P \xrightarrow{\mu} P'$ where μ is either an input or a τ, then $[\![P]\!] \xrightarrow{\mu}\rightarrowtail [\![P']\!]$.*
- *If $P \xrightarrow{\mu} P'$ where μ is neither an input nor a τ, then $[\![P]\!] \xrightarrow{\mu} [\![P']\!]$.*
- *If $[\![P]\!] \xrightarrow{\mu} Q$, where μ is either $a?1\langle b\rangle$ or τ, then there is P' s.t. $P \xrightarrow{\mu} P'$ and $Q \rightarrowtail [\![P']\!]$.*
- *If $[\![P]\!] \xrightarrow{\mu} Q$, where μ is neither an input nor a τ, then there is P' s.t. $P \xrightarrow{\mu} P'$ and $[\![P']\!] = Q$.*

Proof. In Appendix: B.1 □

The correspondence of transitions above encoding is very tight, admittedly because we use a semantics where case analysis does not take a step, otherwise there would be additional steps involved to "perform" the case analysis.

In that case we would lose full abstraction w.r.t strong equivalences as shows this example : Let $X = (a?v.P)|(a?v.P)$ and $Y = a?v.(P|a?v.P)$. In the case of Y the first input at a has to be "fully processed" (i.e. the case analysis must be done) before another input at a can start. In the case of X we can start the second input before the first one is complete.

We would however have weak operational correspondence and full abstraction w.r.t weak equivalences anyway because a case analysis step is deterministic and independent, i.e., case analysis does not interfere with any other term.

Note that we get deadlock- and divergence-freedom as a corollary of the Proposition 3.0.1, so the encoding is "good".

Let $\mathcal{P}$ denote TyCO $\cup\, \pi_{\mathrm{a}}^{V}$ and generalize $\sim$ to this setting.

Proposition 3.0.2. *Let P be a TyCO term. Then $P \sim [\![P]\!]$.*

Proof. In Appendix : B.3. □

The encoding is fully abstract with respect to any behavioral equivalence, again because of Proposition 3.0.1, and also with respect to structural congruence.

Proposition 3.0.3 (Full Abstraction)**.** *Let P_1, P_2 be TyCO terms. Let $\mathcal{R} \in \{\equiv, \sim, \cong\}$. Then $P_1 \mathrel{\mathcal{R}} P_2$ iff $[\![P_1]\!] \mathrel{\mathcal{R}} [\![P_2]\!]$.*

Proof. The proof for $\equiv$ is in appendix : B.4.

The proof for $\sim$ and $\cong$ come from 3.0.2 :

$P \sim [\![P]\!] \cong [\![Q]\!] \sim Q$ implies $P \cong Q$ and $[\![P]\!] \sim P \cong Q \sim [\![Q]\!]$ implies $[\![P]\!] \cong [\![Q]\!]$. The proof for $\sim$ works the same way. □

4 Receptive TyCO

In order to reason formally about an encoding of π_{a}^{V} into TyCO, as we will show in the next section, we build upon the theory of uniform receptiveness, as proposed by Sangiorgi [San99a, San99b]. Receptiveness is a property of a name in a π-calculus term. Two different versions have been studied: *uniform* and *linear* receptiveness. *A name u is uniform in a process P, if (1) at any time P is ready to accept inputs at u and (2) the input offer at u is functional, i.e., all messages received by P at u are applied to the same continuation* [San99a]. A name l is linear in P, if P offers just a single input at l, but it does so immediately. Names that are neither required to be linear or uniform are called *plain* names The main contribution of [San99a] was a receptiveness type system that guarantees the above semantic conditions on uniform and linear names (while putting no constraint at all on plain names) and, moreover, dedicated notions of barbed process equivalences (and labeled characterizations thereof) that only consider process contexts that maintain—by requiring well-typedness—the receptiveness criteria also for the composite process. The resulting notions of equivalence are strictly coarser than the standard notions, because they consider, by definition, fewer contexts in which terms are required to behave the same. Consequently, more terms can be equated, which is very useful in many encodings of the π-calculus literature, and also in our case.

Uniform and linear receptiveness have been studied separately, but the technical development is very similar, and they can be combined in a straightforward manner as exemplified in the application of [San99b].

For clarity, we follow a convention on meta-variables for names, distinguishing the three distinct cases of plain (p), uniform (u), and linear (l) names; if we do not care about receptiveness we use meta-variables for any name (a, b, x, z). Uniform and linear names are also commonly referred to as *receptive* (r) names, as opposed to plain names.

In this section, we transfer the receptiveness theory from π-calculus to TyCO. In doing so, we have to both specialize and generalize the setting. The specialization is due to the fact that TyCO employs asynchronous output only, while the π-calculus of [San99a] builds on synchronous output. The generalization is due to the fact that (i) TyCO includes a richer value language, and (2) Sangiorgi only considered the special cases, where receptive names always carry plain names, while our application requires the transmission of receptive names on receptive names: of the nine theoretically possible combinations of communication, we precisely use "uniform on plain", "uniform on linear", and "linear on uniform", but it is then not much more work to present the receptiveness type system in full generality. Interestingly, the generalization is rather smooth, while the specialization to asynchronous output required a bit more work.

4.1 Receptive Types

Following the above-mentioned convention for plain/uniform/linear names, we assume $\mathbf{N} \overset{\text{def}}{=} \mathbf{N}_\mathtt{p} \oplus \mathbf{N}_\mathtt{u} \oplus \mathbf{N}_\mathtt{l}$. For convenience, if $A \subseteq \mathbf{N}$ then we define $A_\mathtt{u} \overset{\text{def}}{=} A \cap \mathbf{N}_\mathtt{u}$ and $A_\mathtt{l} \overset{\text{def}}{=} A \cap \mathbf{N}_\mathtt{l}$. Additionally, $A_\mathtt{r} \overset{\text{def}}{=} A \cap (\mathbf{N}_\mathtt{l} \cup \mathbf{N}_\mathtt{u})$.

Auxiliary Syntax *Labeled receptive bisimilarity*, as introduced in Definition 4.2.9, builds on the notion of *discreet* processes, which never exhibit a free output of any receptive name. Discreetness is guaranteed by the typability of processes exclusively using rules for *bound output*. This technique is slightly trickier to achieve in a setting where there is no primitive output prefix construct. Therefore, we introduce additional syntactic constructs to the grammar defining TyCO processes

$$\begin{array}{rcll} P & ::= & \dots & \\ & \mid & a!\mathtt{l}(\boldsymbol{\nu} b).P & \text{bound output prefix} \\ & \mid & a \gg b & \text{dynamic link} \end{array}$$

that can be defined within the given calculus as derived operators as follows: bound output $a!\mathtt{l}(\boldsymbol{\nu} b).P_b$ (see Boreale [Bor98] for its first use within an asynchronous calculus) is ever only going to be used with P_b being a (possibly replicated) input at channel b, such that

$$a!\mathtt{l}(\boldsymbol{\nu} b).P_b \overset{\text{def}}{=} (\boldsymbol{\nu} b)\,(\ a!\mathtt{l}\langle b\rangle \mid P_b\ )$$

holds; dynamic links are defined according to Sangiorgi [San96] as

$$a \gg b \overset{\text{def}}{=} \begin{cases} a?^*\{\mathrm{l}_j(x) = b!\mathrm{l_j}(\boldsymbol{\nu} z).z \gg x \mid j \in J\} & \text{if } a \in \mathbf{N}_{\mathtt{u}} \cup \mathbf{N}_{\mathtt{p}} \\ a?\{\mathrm{l}_j(x) = b!\mathrm{l_j}(\boldsymbol{\nu} z).z \gg x \mid j \in J\} & \text{if } a \in \mathbf{N}_{\mathtt{l}} \end{cases}$$

where the definition is "polymorphic" in the labels l_j and the J set (which must match the type of a and b). In this paper, we will use it both with non-branching and branching inputs at a. We introduce dynamic links because of the fact that discreetness must be guaranteed recursively in the case of the transmission of "receptive over receptive" names, while for Sangiorgi [San98] *static links* were sufficient.

Receptiveness Type System Judgments $\Delta_{\mathtt{l}}; \Gamma_{\mathtt{l}}; \Delta_{\mathtt{u}}; \Gamma_{\mathtt{u}} \vdash_{\mathrm{R}} P$, where $\Gamma_{\mathtt{l}} \cup \Delta_{\mathtt{l}} \subseteq \mathbf{N}_{\mathtt{l}}$ and $\Gamma_{\mathtt{u}} \cup \Delta_{\mathtt{u}} \subseteq \mathbf{N}_{\mathtt{u}}$, are such that $\Delta = \Delta_{\mathtt{l}} \cup \Delta_{\mathtt{u}}$ denotes admitted inputs on receptive names, while $\Gamma = \Gamma_{\mathtt{l}} \cup \Gamma_{\mathtt{u}}$ denotes admitted outputs on receptive names, so $\mathrm{fn}(P)_{\mathtt{l}} = \mathrm{fn}(P) \cap \mathbf{N}_{\mathtt{l}} \subseteq \Delta_{\mathtt{l}} \cup \Gamma_{\mathtt{l}}$, $\mathrm{fn}(P)_{\mathtt{u}} = \mathrm{fn}(P) \cap \mathbf{N}_{\mathtt{u}} \subseteq \Delta_{\mathtt{u}} \cup \Gamma_{\mathtt{u}}$, and $\mathrm{fn}(P)_{\mathtt{r}} = \mathrm{fn}(P)_{\mathtt{l}} \cup \mathrm{fn}(P)_{\mathtt{u}}$. We sometimes also abbreviate judgments to $\Delta; \Gamma \vdash_{\mathrm{R}} P$ or even to $\Theta \vdash_{\mathrm{R}} P$. Note that there is some weakening of judgments, both in the position of $\Gamma_{\mathtt{u}}$ and $\Delta_{\mathtt{l}}$, $\Gamma_{\mathtt{l}}$. This is because outputs on uniform names may occur arbitrarily often, even not at all. Weakening on linear names was introduced to handle the case where a linear name was restricted, after a transfer is made over that name the restriction is still there but not the input and the output. By $x \notin \Gamma$, we abbreviate that $x \in \mathbf{N}_{\mathtt{t}}$ implies $x \notin \Gamma_{\mathtt{t}}$ for $\mathtt{t} \in \{\mathtt{l}, \mathtt{u}\}$.

Judgments are generated by the typing rules in Tables 5–7, where the last table contains only rules that are derivable from the others (with a slight exception in the case of (R-Link), as we shall see), if we admit free outputs within terms, i.e., rule (R-Out). We find it convenient to collapse similar rules into one rule schema and to use a case analysis on the kind of name that is restricted on or exchanged in an action rather than let the number of rules explode. Such schemas also allow us to pinpoint the differences between the individual cases. The rules in Table 5 are straightforward: rule (R-Par) requires the disjointness on all Γ/Δ-components but $\Gamma_{\mathtt{u}}$, because there shall be precisely one receiver for any receptive name, and there shall not be more than one output on a linear name. Rule (R-Res) adds the restricted name to both the output and the input set of the respective kind.

The rules in Table 6 concern the recording of increased capabilities (only for use in outputs) of a process when receiving a receptive name, which is

$$\frac{\Delta_{\mathtt{l}} = \Gamma_{\mathtt{l}}}{\Delta_{\mathtt{l}}; \Gamma_{\mathtt{l}}; \emptyset; \Gamma_{\mathtt{u}} \vdash_{\mathrm{R}} \mathbf{0}} \quad (\text{R-Nil})$$

$$\frac{\{a\}_{\mathtt{l}} \cap \{b\}_{\mathtt{l}} = \emptyset \qquad \Gamma_{\mathtt{l}} = \Delta_{\mathtt{l}} \oplus \{a, b\}_{\mathtt{l}} \qquad \{a, b\}_{\mathtt{u}} \subseteq \Gamma_{\mathtt{u}}}{\Delta_{\mathtt{l}}; \Gamma_{\mathtt{l}}; \emptyset; \Gamma_{\mathtt{u}} \vdash_{\mathrm{R}} a!\mathrm{l}\langle b\rangle} \quad (\text{R-Out})$$

$$\frac{\Delta_{1\mathtt{l}}; \Gamma_{1\mathtt{l}}; \Delta_{1\mathtt{u}}; \Gamma_{\mathtt{u}} \vdash_{\mathrm{R}} P_1 \qquad \Delta_{2\mathtt{l}}; \Gamma_{2\mathtt{l}}; \Delta_{2\mathtt{u}}; \Gamma_{\mathtt{u}} \vdash_{\mathrm{R}} P_2}{\Delta_{1\mathtt{l}} \oplus \Delta_{2\mathtt{l}}; \Gamma_{1\mathtt{l}} \oplus \Gamma_{2\mathtt{l}}; \Delta_{1\mathtt{u}} \oplus \Delta_{2\mathtt{u}}; \Gamma_{\mathtt{u}} \vdash_{\mathrm{R}} P_1 | P_2} \quad (\text{R-Par})$$

$$\frac{\begin{array}{llllll} a \notin \Gamma \cup \Delta \\ a \in \mathbf{N}_{\mathtt{p}} \Rightarrow & \Delta_{\mathtt{l}}; & \Gamma_{\mathtt{l}}; & \Delta_{\mathtt{u}}; & \Gamma_{\mathtt{u}} & \vdash_{\mathrm{R}} P \\ a \in \mathbf{N}_{\mathtt{u}} \Rightarrow & \Delta_{\mathtt{l}}; & \Gamma_{\mathtt{l}}; & \Delta_{\mathtt{u}}, a; & \Gamma_{\mathtt{u}}, a & \vdash_{\mathrm{R}} P \\ a \in \mathbf{N}_{\mathtt{l}} \Rightarrow & \Delta_{\mathtt{l}}, a; & \Gamma_{\mathtt{l}}, a; & \Delta_{\mathtt{u}}; & \Gamma_{\mathtt{u}} & \vdash_{\mathrm{R}} P \end{array}}{\Delta_{\mathtt{l}}; \Gamma_{\mathtt{l}}; \Delta_{\mathtt{u}}; \Gamma_{\mathtt{u}} \vdash_{\mathrm{R}} (\boldsymbol{\nu} a)\, P} \quad (\text{R-Res})$$

Table 5: Receptive Types — Basic Rules

complicated by the fact that the reception may itself be *on* some receptive name. The $\Phi_{\mathtt{l}}$ symbol is for permitting weakening on linear names.

Reception on a *plain* name is possible for both input and replication. In contrast, reception on a *linear* or *uniform* name is only possible for input *or* replication, respectively: a linear name could not be linear if it was replicated, and it could not be statically guaranteed to behave uniformly if it was used in a single input. Note that the rules imply the locality property of receptive names, which becomes clear when recalling that a receptive name cannot be activated only after transmission—it must be active from the moment of its "birth".

The rules in Table 7 do not offer many more surprises, because bound output acts very much like a binder itself, which is clear considering that it is essentially a restricted term, where one (output) capability is immediately passed on. Note, however, that the continuation process P_b of our auxiliary notation for bound outputs immediately provides an input capability of appropriate (linear or uniform) kind when required. (In fact, if $b \in \mathbf{N}_{\mathtt{p}}$ or $b \in \mathbf{N}_{\mathtt{u}}$, then with $P_b = b?^*\{\mathrm{l}_j(x_j){=}P_j \mid j{\in}J\}$ the typability premise implies $\Gamma_{\mathtt{l}} = \emptyset$.) The derivability of that rule is in Appendix (B.5).

Finally, concerning the rule (R-Link). Note that, unlike the two other

$$\frac{\begin{array}{c} a \notin \mathbf{N}_{\mathtt{u}} \qquad \Delta_{\mathtt{l}} = \{a\}_{\mathtt{l}} \qquad a \notin \Gamma_{\mathtt{l}} \\ (\forall j{\in}J) \begin{pmatrix} x_j \notin \Gamma \cup \Delta \cup \Phi & & & & \\ x_j \in \mathbf{N}_{\mathtt{p}} & \Longrightarrow & \emptyset; \Gamma_{\mathtt{l}}; & \emptyset; \Gamma_{\mathtt{u}} & \vdash_{\mathrm{R}} P_j \\ x_j \in \mathbf{N}_{\mathtt{u}} & \Longrightarrow & \emptyset; \Gamma_{\mathtt{l}}; & \emptyset; \Gamma_{\mathtt{u}}, x_j & \vdash_{\mathrm{R}} P_j \\ x_j \in \mathbf{N}_{\mathtt{l}} & \Longrightarrow & \emptyset; \Gamma_{\mathtt{l}}, x_j; & \emptyset; \Gamma_{\mathtt{u}} & \vdash_{\mathrm{R}} P_j \end{pmatrix} \end{array}}{\Delta_{\mathtt{l}} \oplus \Phi_{\mathtt{l}}; \Gamma_{\mathtt{l}} \oplus \Phi_{\mathtt{l}}; \emptyset; \Gamma_{\mathtt{u}} \vdash_{\mathrm{R}} a?\{\mathrm{l}_j(x_j){=}P_j \mid j{\in}J\}} \text{ (R-I{\scriptsize NP})}$$

$$\frac{\begin{array}{c} a \notin \mathbf{N}_{\mathtt{l}} \qquad \Delta_{\mathtt{l}} = \Gamma_{\mathtt{l}} \qquad \{a\}_{\mathtt{u}} = \Delta_{\mathtt{u}} \\ (\forall j{\in}J) \begin{pmatrix} x_j \notin \Gamma \cup \Delta & & & & \\ x_j \in \mathbf{N}_{\mathtt{p}} & \Longrightarrow & \emptyset; \emptyset; & \emptyset; \Gamma_{\mathtt{u}} & \vdash_{\mathrm{R}} P_j \\ x_j \in \mathbf{N}_{\mathtt{u}} & \Longrightarrow & \emptyset; \emptyset; & \emptyset; \Gamma_{\mathtt{u}}, x_j & \vdash_{\mathrm{R}} P_j \\ x_j \in \mathbf{N}_{\mathtt{l}} & \Longrightarrow & \emptyset; \emptyset, x_j; & \emptyset; \Gamma_{\mathtt{u}} & \vdash_{\mathrm{R}} P_j \end{pmatrix} \end{array}}{\Delta_{\mathtt{l}}; \Gamma_{\mathtt{l}}; \Delta_{\mathtt{u}}; \Gamma_{\mathtt{u}} \vdash_{\mathrm{R}} a?^*\{\mathrm{l}_j(x_j){=}P_j \mid j{\in}J\}} \text{ (R-R{\scriptsize EP})}$$

Table 6: Receptive Types — Input Rules

rules in Table 7, it is not entirely derivable from the other rules because of the recursive nature of dynamic links. Actually, as shown in more details in the proof (Section B.6) the part that is undecidable is $\Gamma_{\mathtt{l}} = \{b\}_{\mathtt{l}}$ so we take it as a *rule*. it is also easy to show that the admission of the case $b \in \mathbf{N}_{\mathtt{l}} \land a \notin \mathbf{N}_{\mathtt{l}}$ would not be derivable. Note that we only intend to prevent the free output of receptive names, so we add rule (R-POUT), a special case of (R-OUT) for the free output of plain names.

Lemma 4.1.1. *Let* $\Delta_{\mathtt{l}}; \Gamma_{\mathtt{l}}; \Delta_{\mathtt{u}}; \Gamma_{\mathtt{u}} \vdash_{\mathrm{R}} P$.

1. (Weakening) *If* $\Phi \cap \mathrm{bn}(P) = \emptyset$, *then* $\Delta_{\mathtt{l}} \oplus \Phi_{\mathtt{l}}; \Gamma_{\mathtt{l}} \oplus \Phi_{\mathtt{l}}; \Delta_{\mathtt{u}}; \Gamma_{\mathtt{u}} \oplus \Phi_{\mathtt{u}} \vdash_{\mathrm{R}} P$.
2. (Strengthening) *then* $\Delta \cap \mathrm{fn}(P); \Gamma \cap \mathrm{fn}(P) \vdash_{\mathrm{R}} P$.

Proof. In Appendix : B.7

□

Lemma 4.1.2 (Congruence)**.** *If* $\Theta_1 \vdash_{\mathrm{R}} P$ *and* $P \equiv Q$ *then there is* Θ_2 *with* $\Theta_2 \vdash_{\mathrm{R}} P, Q$.

Lemma 4.1.3 (Free names)**.** *If* $\Delta_{\mathtt{l}}; \Gamma_{\mathtt{l}}; \Delta_{\mathtt{u}}; \Gamma_{\mathtt{u}} \vdash_{\mathrm{R}} P$, *then* $\mathrm{fn}(P)_{\mathtt{u}} \subseteq \Gamma_{\mathtt{u}} \cup \Delta_{\mathtt{u}}$ *and* $\mathrm{fn}(P)_{\mathtt{l}} \subseteq \Gamma_{\mathtt{l}} \cup \Delta_{\mathtt{l}}$ *and* $\Delta_{\mathtt{u}} \subseteq \mathrm{fn}(P)_{\mathtt{u}}$.

$$\frac{b \in \mathbf{N}_\mathtt{p} \qquad \Gamma_\mathtt{l} = \Delta_\mathtt{l} \oplus \{a\}_\mathtt{l} \qquad \{a\}_\mathtt{u} \subseteq \Gamma_\mathtt{u}}{\Delta_\mathtt{l}; \Gamma_\mathtt{l}; \emptyset; \Gamma_\mathtt{u} \vdash_\mathrm{R} a!\mathtt{l}\langle b\rangle} \;(\text{R-Pout})$$

$$\frac{\begin{array}{c} b \notin \Gamma \cup \Phi_\mathtt{l} \qquad \Gamma_\mathtt{l} = \Delta_\mathtt{l} \oplus \{a\}_\mathtt{l} \qquad \{a\}_\mathtt{u} \subseteq \Gamma_\mathtt{u} \\ \left(\begin{array}{lllll} b \in \mathbf{N}_\mathtt{p} \implies \emptyset; & \Gamma_\mathtt{l}\backslash\{a\}_\mathtt{l}; & \emptyset; & \Gamma_\mathtt{u} & \vdash_\mathrm{R} P_b \\ b \in \mathbf{N}_\mathtt{u} \implies \emptyset; & \Gamma_\mathtt{l}\backslash\{a\}_\mathtt{l}; & \emptyset, b; & \Gamma_\mathtt{u}, b & \vdash_\mathrm{R} P_b \\ b \in \mathbf{N}_\mathtt{l} \implies \emptyset, b; & \Gamma_\mathtt{l}\backslash\{a\}_\mathtt{l}; & \emptyset; & \Gamma_\mathtt{u} & \vdash_\mathrm{R} P_b \end{array}\right) \end{array}}{\Phi_\mathtt{l}; \Gamma_\mathtt{l} \oplus \Phi_\mathtt{l}; \emptyset; \Gamma_\mathtt{u} \vdash_\mathrm{R} a!\mathtt{l}(\boldsymbol{\nu} b).P_b} \;(\text{R-Bout})$$

$$\frac{b \in \mathbf{N}_\mathtt{l} \Rightarrow a \in \mathbf{N}_\mathtt{l} \qquad \{b\}_\mathtt{u} \subseteq \Gamma_\mathtt{u}}{\{a\}_\mathtt{l} \oplus \Phi_\mathtt{l}; \{b\}_\mathtt{l} \oplus \Phi_\mathtt{l}; \{a\}_\mathtt{u}; \Gamma_\mathtt{u} \vdash_\mathrm{R} a \gg b} \;(\text{R-Link})$$

Table 7: Receptive Types — Discreetness Rules

Proof. In Appendix : B.7

□

Lemma 4.1.4 (Type substitution). *Let* $\Delta_\mathtt{l}; \Gamma_\mathtt{l}; \Delta_\mathtt{u}; \Gamma_\mathtt{u} \vdash_\mathrm{R} P$.

1. $\Delta; \Gamma \vdash_\mathrm{R} P\{{}^{p}\!/\!_{p'}\}$.
2. *If* $r \notin \Delta \cup \Gamma$ *and* $r' \in \Gamma - \Delta$, *then* $\Delta; (\Gamma - r'), r \vdash_\mathrm{R} P\{{}^{r}\!/\!_{r'}\}$.
3. *If* $r \notin \Delta \cup \Gamma$ *and* $r' \in \Delta \cap \Gamma$, *then* $(\Delta - r'), r; (\Gamma - r'), r \vdash_\mathrm{R} P\{{}^{r}\!/\!_{r'}\}$.

The following proposition needs to respect the type, in contrast to Davide's setting, where receptive always carries plain

Proposition 4.1.5 (Input receptiveness). *Let* $\Delta; \Gamma \vdash_\mathrm{R} P$.
If $r \in \Delta_\mathtt{l} \cap \mathrm{fn}(P)$ *or* $r \in \Delta_\mathtt{u}$, *then* $\forall b \forall \mathtt{l}$ *"with type matching the one of* r*" and* $b \notin \Gamma_\mathtt{l}$: $\exists Q : P \xrightarrow{r?\mathtt{l}\langle b\rangle} Q$.

Proposition 4.1.6 (Type soundness). *Let* $\Delta_\mathtt{l}; \Gamma_\mathtt{l}; \Delta_\mathtt{u}; \Gamma_\mathtt{u} \vdash_\mathrm{R} P$.

1. *If* $P \xrightarrow{a?\mathtt{l}\langle b\rangle} Q$ *with* $\{b\}_\mathtt{l} \cap \Gamma_\mathtt{l} = \emptyset$ *and* b*'s type matching the one of* a,
   then $\{a\}_\mathtt{u} \subseteq \Delta_\mathtt{u}$, $\{a\}_\mathtt{l} \subseteq \Delta_\mathtt{l}$,
   and $\Delta_\mathtt{l}\backslash\{a\}_\mathtt{l}; \Gamma_\mathtt{l} \cup \{b\}_\mathtt{l}; \Delta_\mathtt{u}; \Gamma_\mathtt{u} \cup \{b\}_\mathtt{u} \vdash_\mathrm{R} Q$.
2. *If* $P \xrightarrow{a!\mathtt{l}\langle b\rangle} Q$,
   then $\{a, b\}_\mathtt{u} \subseteq \Gamma_\mathtt{u}$, $\{a, b\}_\mathtt{l} \subseteq \Gamma_\mathtt{l}$,
   and $\Delta_\mathtt{l}; \Gamma_\mathtt{l}\backslash\{a, b\}_\mathtt{l}; \Delta_\mathtt{u}; \Gamma_\mathtt{u} \vdash_\mathrm{R} Q$.

3. *If* $P \xrightarrow{(\boldsymbol{\nu} b)\, a!1\langle b\rangle} Q$,
   Let $\Phi_1 = \Gamma_1 \cap \{b\}$. *Then*
   $\Delta_1 \setminus \Phi_1; \Gamma_1 \setminus \Phi_1; \Delta_u; \Gamma_u \vdash_R P$,
   $\{a\}_u \subseteq \Gamma_u$, $\{b\}_u \cap \Gamma_u = \emptyset$, $\{a\}_1 \subseteq \Gamma_1$
   and $\Delta_1 \setminus \Phi_1 \cup \{b\}_1; \Gamma_1 \setminus \Phi_1 \setminus \{a\}_1; \Delta_u \cup \{b\}_u; \Gamma_u \cup \{b\}_u \vdash_R Q$.
4. *If* $P \xrightarrow{\tau} Q$, *then either* $\Delta_1; \Gamma_1; \Delta_u; \Gamma_u \vdash_R Q$.
   or $\exists l \in \Delta_1 \cap \Gamma_1$ *with* $\Delta_1 - l; \Gamma_1 - l; \Delta_u; \Gamma_u \vdash_R Q$.

Proof. In Appendix : B.8

□

Some comments on the soundness:

- In case of free input : the "$\{b\}_1 \cap \Gamma_1 = \emptyset$" condition will always be respected by a well-typed observer.
- In the case of bound output, the b name might be present in Δ_1 and Γ_1 by weakening, so it must be removed from both to allow putting it back in Δ_1
- in the case of free input, b may be already occurring in Δ_1

4.2 Receptive Congruence

Barbed congruence under receptiveness, as introduced by Sangiorgi [San99b], is defined as barbed bisimulation under *completing contexts*. A process is *complete* if it includes itself the receivers required for all outputs on or of receptive names such that no receptive name can occur free in output position.

Our completeness notion merges [San99a]'s completeness on linear names and semi-completeness on uniform names.

Definition 4.2.1 (Completeness). *A process P is* complete *if there are Δ_1, Δ_u, and Γ_u such that $\Gamma_u \subseteq \Delta_u$ and $\Delta_1; \emptyset; \Delta_u; \Gamma_u \vdash_R P$. A context $C[\cdot]$ is complete on $(\Delta; \Gamma)$, if $C[P]$ is complete for all P with $\Delta; \Gamma \vdash_R P$.*

Lemma 4.2.2 (Completeness preservation). *Let P be complete. Then:*

1. *any $Q \equiv P$ is complete;*
2. *if $P \xrightarrow{\mu} Q$ with μ being an output, or a τ-action, or an input with plain object, then Q is complete;*
3. *if $P \xrightarrow{a?1\langle r\rangle} Q$ with r fresh and v plain, then $(\boldsymbol{\nu} r)\,(r \gg v \mid Q)$ is complete.*

Definition 4.2.3 (Receptive Barbed Congruence). *Let $\Delta;\Gamma \vdash_{\mathrm{R}} P,Q$. Then, P and Q are* receptive barbed congruent *on $(\Delta;\Gamma)$, written $P \cong_{\mathrm{R}} Q$, if for each context $C[\cdot]$, which is complete on $(\Delta;\Gamma)$, it holds that $C[P] \stackrel{\bullet}{\approx} C[Q]$.*

Labeled bisimilarity characterizes barbed congruence under the proviso that there are no free outputs of receptive names. We can guarantee that by means of receptive typability ignoring the rule for free output.

Definition 4.2.4 (Discreetness). *A process P is called* discreet *if there are Γ and Δ with $\Delta;\Gamma \vdash_{\mathrm{R}} P$ without using rule* (R-Out)*, but instead using the additional rules in Table 7.*

Lemma 4.2.5 (Discreetness Preservation). *Let P be discreet. Then:*

1. *if $P \xrightarrow{\mu} Q$ with μ being an output, or a τ-action, or an input with plain object, then Q is discreet;*
2. *if $P \xrightarrow{a?1\langle r\rangle} Q$ with r fresh, then Q is discreet.*
3. *any $Q \equiv P$ is discreet.*

The notion of receptive labeled bisimulation is now, with the adapted definition of discreetness, a mere replay of Sangiorgi's [San99a].

Definition 4.2.6 (Weak Receptive Bisimulation). *A symmetric relation $\mathcal{R}$ on complete discreet processes is a* weak receptive bisimulation*, if $P \mathrel{\mathcal{R}} Q$ implies:*

1. *if $P \xrightarrow{\mu} P'$ with* $\mathrm{bn}(\mu)$ *fresh, and μ is an* output with plain subject *or an* input with plain object*, then there is Q' with $Q \xRightarrow{\mu} Q'$ and $P' \mathrel{\mathcal{R}} Q'$.*
2. *if $P \xrightarrow{\tau} P'$, then there is Q' with $Q \Rightarrow Q'$ and $P' \mathrel{\mathcal{R}} Q'$.*
3. *if $P \xrightarrow{a?1\langle r\rangle} P'$ with r fresh, then, for some plain and fresh name t, there are Q' and Q'' with*
   (a) *$Q \xRightarrow{a?1\langle r\rangle} Q'$,*
   (b) *$(\boldsymbol{\nu} r)\,(\, r \gg t \mid Q'\,) \Rightarrow Q''$, and*
   (c) *$(\boldsymbol{\nu} r)\,(\, r \gg t \mid P'\,) \mathrel{\mathcal{R}} Q''$.*

Note that we do not have to consider outputs with receptive subjects, because—by completeness—the receivers on such names are within the term, so a well-typed observer will not be able to receive on them itself. Moreover, with respect to receptiveness, output with receptive objects are no problem, because—by discreetness—they are always bound and provide the receiver within the term itself immediately afterwards. Note further that the name t plays the role of Sangiorgi's trigger names [San99a].

Here are definitions of the stronger equivalences :

Definition 4.2.7 (Receptive Expansion)**.** *A relation $\mathcal{R}$ is a receptive expansion if $P \mathcal{R} Q$ implies :*

1. *$P \xrightarrow{\mu} P'$ with μ as in point 1. in the above definition implies there is Q' such that $Q \overset{\mu}{\Longrightarrow} Q'$ and $P' \mathcal{R} Q'$.*
2. *$P \xrightarrow{\tau} P'$ implies there is Q' such that $Q \Rightarrow Q'$ and $P' \mathcal{R} Q'$.*
3. *$P \xrightarrow{\mu} P'$ with μ as in point 3. in the above definition implies there are Q' and Q'' such that $Q \overset{\mu}{\Longrightarrow} Q'$, $(\boldsymbol{\nu} r)\,(\, r \gg t \mid Q' \,) \Rightarrow Q''$ and $(\boldsymbol{\nu} r)\,(\, r \gg t \mid P' \,) \mathcal{R} Q''$, for some fresh and plain t.*
4. *$Q \xrightarrow{\mu} Q'$ with μ as in point 1. in the above definition implies there is P' such that $P \xrightarrow{\mu} P'$ and $P' \mathcal{R} Q'$.*
5. *$Q \xrightarrow{\tau} Q'$ implies either $P \mathcal{R} Q'$ or there is P' such that $P \xrightarrow{\tau} P'$ with $P' \mathcal{R} Q'$.*
6. *$Q \xrightarrow{\mu} Q'$ with μ as in point 3. in the above definition implies there is P' such that $P \xrightarrow{\mu} P'$ and $(\boldsymbol{\nu} r)\,(\, r \gg t \mid P' \,) \mathcal{R} (\boldsymbol{\nu} r)\,(\, r \gg t \mid Q' \,)$, for some fresh and plain t.*

Definition 4.2.8 (Strong Receptive Bisimulation)**.** *The corresponding strong definition is as follows :*

A symmetric relation $\mathcal{R}$ on complete discreet processes is a strong receptive bisimulation, *if $P \mathcal{R} Q$ implies:*

1. *if $P \xrightarrow{\mu} P'$ with $\mathrm{bn}(\mu)$ fresh, and μ is an output with plain subject, τ or an input with plain object, then there is Q' with $Q \xrightarrow{\mu} Q'$ and $P' \mathcal{R} Q'$.*
2. *if $P \xrightarrow{a?1\langle r\rangle} P'$ with r fresh, then, for some plain and fresh name t, there is Q' with $Q \xrightarrow{a?1\langle r\rangle} Q'$ and $(\boldsymbol{\nu} r)\,(\, r \gg t \mid P' \,) \mathcal{R} (\boldsymbol{\nu} r)\,(\, r \gg t \mid Q' \,)$.*

Definition 4.2.9 (Labeled Receptive Bisimilarity / Expansion)**.** *P and Q are* strong/weak receptive bisimilar, *written $P \sim_{\mathrm{R}} Q$ or $P \approx_{\mathrm{R}} Q$, respectively, if there is a strong/weak bisimulation $\mathcal{R}$ such that $P \mathcal{R} Q$.*

Q receptive-expands P*, written* $Q \gtrsim_{\mathrm{R}} P$ *if there is a receptive expansion* $\mathcal{R}$ *such that* $P \mathrel{\mathcal{R}} Q$*.*

- labeled vs. barbed

Lemma 4.2.10 (Congruence)**.** *Consider* P *and* Q *discreet and complete.*

Then, with $C[\cdot]$ *a context such that* $C[P]$ *and* $C[Q]$ *are discreet and complete and* $\mathcal{R} \in \{\sim_{\mathrm{R}}, \gtrsim_{\mathrm{R}}, \approx_{\mathrm{R}}\}$ *:*

- $C[P] \mathrel{\mathcal{R}} C[Q]$ *iff* $P \mathrel{\mathcal{R}} Q$

Proposition 4.2.11 (Replication)**.** *Having* $u \in \mathbf{N}_{\mathtt{u}}$*,* $v \in \mathbf{N}_{\mathtt{p}}$ *and* P *and* Q *discreet such that*

$\Delta_{P\mathtt{1}}; \emptyset; \Delta_{P\mathtt{u}}; \Gamma_{\mathtt{u}}, u \vdash_{\mathrm{R}} P$ *and* $\Delta_{Q\mathtt{1}}; \emptyset; \Delta_{Q\mathtt{u}}; \Gamma_{\mathtt{u}}, u \vdash_{\mathrm{R}} Q$ *:*

If the following conditions (which are necessary and sufficient for the processes of the below pair to be complete) hold :

- $\Gamma_{\mathtt{u}} \subseteq \Delta_{P\mathtt{u}} \oplus \Delta_{Q\mathtt{u}}$
- $\Delta_{P\mathtt{1}} \cap \Delta_{Q\mathtt{1}} = \emptyset$ *and* $u \notin \Delta_{P\mathtt{u}} \cup \Delta_{Q\mathtt{u}}$

Then:

$$(\boldsymbol{\nu} u)\,(\, u \gg v \mid P \mid Q\,) \sim_{\mathrm{R}} (\boldsymbol{\nu} u)\,(\, u \gg v \mid P\,) \mid (\boldsymbol{\nu} u)\,(\, u \gg v \mid Q\,)$$

Proof. In Appendix : B.9

□

5 Translating π_{a}^{V} into TyCO: Nested Variants as non-nested variants

We present a generalization of an encoding of Lπ (Local π), the asynchronous π-calculus with output-only mobility [Mer00], into πI, the π-calculus with internal-only mobility [San96], that has first been studied by Boreale [Bor98], then by Merro and Sangiorgi [MS98]. Here, the source language is Local π_{a}^{V}, i.e., Lπ extended with nested labeled values, while the target language is TyCO, as we know by § 3 for well-typed settings, is just π_{a}^{V} with only flat variants. In fact, our encoding uses only the subset of TyCO that is both local and internal, which is a promising setting, because it has been shown

$$\begin{aligned}
[\![\,[T]\,]\!] &\stackrel{\text{def}}{=} [\{\mathrm{c}{:}[\![\,T\,]\!]\}] \\
[\![\,\{\mathrm{l}_j{:}T_j \mid j{\in}J\}\,]\!] &\stackrel{\text{def}}{=} [\{\mathrm{d}{:}[\{\mathrm{l}_j{:}[\![\,T_j\,]\!] \mid j{\in}J\}]\}] \\
[\![\,\mu X.T\,]\!] &\stackrel{\text{def}}{=} \mu X.[\![\,T\,]\!] \\
[\![\,X\,]\!] &\stackrel{\text{def}}{=} X
\end{aligned}$$

Table 8: Encoding Nested Variant Types

that only when using this restricted target language the above encoding is known to be fully abstract with respect to barbed congruence [MS98].

We can rather straightforwardly enhance our encoding to also encompass non-local terms (Section 5.3). The reason for considering local calculi only is that we need a stronger theory, the theory of receptiveness[San99a, San99b], to prove nice results about non-local terms.

5.1 Translating Types

When translating the free nesting of π_{a}^{V} into the alternating nesting of TyCO, additional 'levels' must be introduced in order to conform to the alternation.

Each type after translation is a channel containing a variant type with a single label that indicate the kind of original type.

We introduce two labels, c and d, for respectively channel types and variant types.

The encoding $[\![\ ]\!]$ of Table 8 translates π_{a}^{V}-types into TyCO-types. The first clause shows how to translate a channel type (with a c layer and then we recurse on the inner type) The second clause shows how to translate a variant type (with one additional d layer and then we recurse on the type expected by each label). This additional d level illustrates the protocol used by the encoding for accessing an encoded variant value :

A variant value (say $\mathrm{l}\langle v\rangle$) is represented by a name that can receive d messages, which will be answered by a message with the highest level label (l) and a new name representing v, the labelled value.

Recursive types and the respective variables are translated homomorphically.

$$\begin{array}{rcl}
[\![\, a!v \,]\!] & \stackrel{\text{def}}{=} & a!\mathrm{c}(\boldsymbol{\nu}u).[\![\, v \,]\!]_u \\
[\![\, b \,]\!]_u & \stackrel{\text{def}}{=} & u \gg b \\
[\![\, \mathrm{l}\langle v\rangle \,]\!]_u & \stackrel{\text{def}}{=} & u?^*\{\mathrm{d}(r){=}r!\mathrm{l}(\boldsymbol{\nu}u').[\![\, v \,]\!]_{u'}\} \\
[\![\, a?(x).P \,]\!] & \stackrel{\text{def}}{=} & a?\{\mathrm{c}(x){=}[\![\, P \,]\!]\} \\
[\![\, a?^*(x).P \,]\!] & \stackrel{\text{def}}{=} & a?^*\{\mathrm{c}(x){=}[\![\, P \,]\!]\} \\
[\![\, \mathsf{case}\; v \;\mathsf{of}\; \{\mathrm{l}_j(x_j){=}P_j \;\mid\; j{\in}J\} \,]\!] & \stackrel{\text{def}}{=} & (\boldsymbol{\nu}u)\;\big(\; [\![\, v \,]\!]_u \mid \\
 & & u!\mathrm{d}(\boldsymbol{\nu}r).r?\{\mathrm{l}_j(x_j){=}[\![\, P_j \,]\!] \;\mid\; j{\in}J\}\;\big)
\end{array}$$

Table 9: Encoding Nested Variants — Local Source Language

5.2 Translating Local Terms

We recall the idea of encoding the higher-order π-calculus into first-order π-calculus [San93], where values are not transmitted themselves, but only private references to them. We apply the same idea here in that complex values—variants—are not transmitted themselves, but only private references to them. (In fact, already the encoding Lπ $\rightarrow$ πI, which we intend to generalize here to labeled values, builds on that idea.)

The clauses in Table 9 encode nested variants into flat variants: it generalizes the previous ones for Lπ by the encoding of values, using label c : if it is a name, then mere forwarders are created, if it is a variant, then a local (receptive) resource is created instead that follows the protocol of stepwise variant decomposition, using label d. The encoding generates forwarders for variants with only one single label (in TyCO-terminology: single-method objects): $\mathrm{l} = \mathrm{d}$. The reply to a d-request, however, will go through a branching dynamic link. In resources representing labeled values, each accepted d-request creates a new resource for the embedded value of the next layer. The advantage over shared resources is for reasoning purposes: no extrusion of the sub-resource has to be considered for subsequent d-access to the same layer.

Example 5.2.1. *This one drove us to want a single translation clause for the case of case, instead of two different ones for names and variants. Let us keep it for some time in the paper, just for reference.*

Let $\Sigma \vdash_{\mathrm{S}} P, Q$ for

$$\begin{aligned} P &\overset{\text{def}}{=} a?(x).b!x \\ Q &\overset{\text{def}}{=} a?(x).\mathsf{case}\, x\, \mathsf{of}\, \{\mathrm{l}(z){=}b!\mathrm{l}\langle z\rangle\} \end{aligned}$$

Then $P \approx Q$, because after reception of any well-typed (and closed w.r.t. variant variables) value v, the case construct can be eliminated by means of structural congruence, freeing the output of precisely the previously received v.

However, $[\![P]\!] \not\approx [\![Q]\!]$, since

$$\begin{aligned} [\![P]\!] &\xrightarrow{a?v} \downarrow_{b!} \\ [\![Q]\!] &\xrightarrow{a?v} \downarrow_{v!} \end{aligned}$$

(where v can only be a name) would be the only two barbs after input of v.

Proposition 5.2.2. *If $\Sigma \vdash_{\mathrm{S}} P$ then*

1. *$[\![\Sigma]\!] \vdash_{\mathrm{S}} [\![P]\!]$,*
2. *$\emptyset; \emptyset; \emptyset; \emptyset \vdash_{\mathrm{R}} [\![P]\!]$, and*
3. *$[\![P]\!]$ is discreet and complete.*

Proof. 1. in Appendix: B.10

2. We first prove $\emptyset; \emptyset; \{u\}; \{u\} \vdash_{\mathrm{R}} [\![v]\!]_u$ by induction on v and then we see that each rule of the encoding produces a process that can be typed under $\emptyset \vdash_{\mathrm{R}} [\![P]\!]$.
3. The completeness is a direct consequence of point 2 and its proof didn't use (R-Out) so we have discreetness as well.

□

Lemma 5.2.3. *Let P and Q be two processes. Then $P \equiv Q$ iff $[\![P]\!] \equiv [\![Q]\!]$.*

Proposition 5.2.4 (Replication)**.** *Having $u \in \mathbf{N_u}$ and P, Q discreet with* $\mathrm{fn}(P)_{\mathtt{r}} \cup \mathrm{fn}(Q)_{\mathtt{r}} \subseteq \{u\}$

$$(\boldsymbol{\nu} u)\,(\,[\![v]\!]_u \mid P \mid Q\,) \sim_{\mathrm{R}} (\boldsymbol{\nu} u)\,(\,[\![v]\!]_u \mid P\,) \mid (\boldsymbol{\nu} u)\,(\,[\![v]\!]_u \mid Q\,)$$

Proof. The proof works the the same way as 4.2.11, from the fact that the only observable input of $[\![v]\!]_u$ is at u and that only output it does is either at a name transmitted to u or at $\mathrm{n}(v)$ (similarly to $u \gg t$. The behaviour of $[\![v]\!]_u$ is given in more details in lemma B.14.1)

□

In order to deal with input- and τ-transitions in the operational correspondence, we need a substitution lemma that relates those actions in the source and target languages, respectively. As closure requirement, we impose the natural constraint that substituting entities do not contain free variant variables.

Note that substitution, operational correspondence and full abstraction only work on a subset of π_{a}^{V} processes, as will be explained in next section in more detail.

Lemma 5.2.5 (Substitution)**.** *Let a and u be different. Let P be a process that does not input on x.*

1. $(\boldsymbol{\nu}u)\,(\, a \gg u \mid [\![\, v \,]\!]_u \,) \gtrsim_{\mathrm{R}} [\![\, v \,]\!]_a$
2. $(\boldsymbol{\nu}u)\,(\, [\![\, P \,]\!]\{^{u}\!/\!_{x}\} \mid [\![\, b \,]\!]_u \,) \gtrsim_{\mathrm{R}} [\![\, P\{^{b}\!/\!_{x}\} \,]\!]$

Proof. 1. Exhibit the relation up to context and up to expansion. (appendix B.11)

2. By induction on the structure of P (appendix B.12)

□

Proposition 5.2.6 (Operational correspondence)**.** *Let P be a π_{a}^{V}-term that doesn't receive on received names and that doesn't contain a $a!v \mid a?(x)$ pair, with $\Sigma \vdash_{\mathrm{S}} P$.*

1. *If $P \rightarrowtail P'$ then $[\![\, P \,]\!] \xrightarrow{\tau}\xrightarrow{\tau} Q \gtrsim_{\mathrm{R}} [\![\, P' \,]\!]$.*
2. *Let $P \xrightarrow{\mu} P'$.*
   (a) *If $\mu = a!v$,*
   then $[\![\, P \,]\!] \xRightarrow{a!\mathrm{c}\,(\boldsymbol{\nu}u)} Q$
   with $\emptyset; \emptyset; \{u\}; \{u\} \vdash_{\mathrm{R}} Q \gtrsim_{\mathrm{R}} [\![\, P' \,]\!] \mid [\![\, v \,]\!]_u$.
   (b) *If $\mu = (\boldsymbol{\nu}b)\, a!v$,*
   then $[\![\, P \,]\!] \xRightarrow{a!\mathrm{c}\,(\boldsymbol{\nu}u)} Q$
   with $\emptyset; \emptyset; \{u\}; \{u\} \vdash_{\mathrm{R}} Q \gtrsim_{\mathrm{R}} (\boldsymbol{\nu}b)\,(\, [\![\, P' \,]\!] \mid [\![\, v \,]\!]_u \,)$.
   (c) *If $\mu = a?v$ ($\mathrm{n}(v)$ fresh),*
   then $[\![\, P \,]\!] \xRightarrow{a?\mathrm{c}\,\langle u \rangle} Q$
   with $\emptyset; \emptyset; \emptyset; \emptyset \vdash_{\mathrm{R}} (\boldsymbol{\nu}u)\,(\, Q \mid [\![\, v \,]\!]_u \,) \gtrsim_{\mathrm{R}} [\![\, P' \,]\!]$.
   (d) *If $\mu = \tau$,*
   then $[\![\, P \,]\!] \xRightarrow{\tau} Q$
   with $\emptyset; \emptyset; \emptyset; \emptyset \vdash_{\mathrm{R}} Q \gtrsim_{\mathrm{R}} [\![\, P' \,]\!]$.

3. Let $[\![P]\!] \xrightarrow{\mu} Q$*.*
Then μ *is one of the following :*

(a) If $\mu = a!\mathrm{c}(\boldsymbol{\nu} u)$*, then* *<u>either</u>*

i. $P \xrightarrow{a!v} P'$*, for some* v *with* $\emptyset;\emptyset;\{u\};\{u\} \vdash_{\mathrm{R}} Q \equiv [\![P']\!] \mid [\![v]\!]_u$*, or*

ii. $P \xrightarrow{(\boldsymbol{\nu} b)\, a!v} P'$*, for some* b *and* v *with* $\emptyset;\emptyset;\{u\};\{u\} \vdash_{\mathrm{R}} Q \equiv (\boldsymbol{\nu} b)(\, [\![P']\!] \mid [\![v]\!]_u \,)$*.*

(b) If $\mu = a?\mathrm{c}\langle u \rangle$*, then* $P \xrightarrow{a?v} P'$*, for some* v *with* $\emptyset;\emptyset;\emptyset;\emptyset \vdash_{\mathrm{R}} (\boldsymbol{\nu} u)(\, Q \mid [\![v]\!]_u \,) \gtrsim_{\mathrm{R}} [\![P']\!]$*,*

(c) If $\mu = \tau$ *then either*

i. $P \xrightarrow{\tau} P'$ *with* $\emptyset;\emptyset;\emptyset;\emptyset \vdash_{\mathrm{R}} Q \gtrsim_{\mathrm{R}} [\![P']\!]$*, or*

ii. $P \rightarrowtail P'$ *and* $Q \xrightarrow{\tau} Q'$ *with* $Q' \gtrsim_{\mathrm{R}} [\![P']\!]$*.*

Proof. In Appendix : B.13 □

Theorem 5.2.7 (Full Abstraction)**.** *Let* $\Sigma \vdash_{\mathrm{S}} P_1, P_2$ *be processes of the* π_{a}^{V} *subset mentioned above.*

Then $P_1 \approx P_2$ *iff* $[\![P_1]\!] \approx_{\mathrm{R}} [\![P_2]\!]$*.*

Proof. In Appendix : B.14 □

5.3 Translating Non-Local Terms

As briefly told in the previous section, a problem of the above encoding is that the substitution lemma is only valid for dealing with translations of local terms in the source language. For example, if we want to match the input transition (with all possible occurrences of the bound name x in input subject and output subject and object position)

$$Q = d?(x).(\, \underbrace{a!x \mid x!a \mid x?(y).\mathbf{0}}_{P} \,) \xrightarrow{d?b} P\{^{b}\!/\!_{x}\}$$

by its translation

$$\begin{array}{rcl} [\![Q]\!] = & \xrightarrow{d?u} & [\![P]\!]\{^{u}\!/\!_{x}\} \\ & = & (\, [\![a!x]\!] \mid [\![x!a]\!] \mid [\![x?(y).\mathbf{0}]\!] \,)\{^{u}\!/\!_{x}\} \\ & = & (\, [\![a!x]\!] \mid [\![x!a]\!] \mid x?\{\mathrm{c}(y){=}[\![\mathbf{0}]\!]\} \,)\{^{u}\!/\!_{x}\} \end{array}$$

then we observe that the substitution would create a receiver at u in addition to the uniform one imposed by $[\![\, b \,]\!]_u$ in the above substitution lemma—thus breaking uniform receptiveness at u. This situation can only arise in cases that terms may receive on received names, thus in non-local terms.

Another problem is that the operational semantics only allow transmission of bound names, i.e. $a!b \mid a?(x).P$ does not allow transmitting b over a because it is not restricted. However, because the encoding creates only bound outputs, the above expression gets translated to

$a!\mathrm{c}(\boldsymbol{\nu} u).[\![\, b \,]\!]_u \mid a?\{\mathrm{c}(x)=[\![\, P \,]\!]\}$

which allows the transfer to take place.

In order to accommodate these problems of translating non-local terms, we will need a slight variant of the previous encoding.

6 Conclusion

As mentionned in the introduction, this whole paper was written with a rule for free-comm. missing in the operational semantics. It seems to preserve all propositions and lemma correct, and if this is the case then the additional condition on Proposition 5.2.6 can be removed, so we should get full abstraction on $\mathrm{L}\pi_{\mathrm{a}}^{V}$ (Local π_{a}^{V}) processes.

In the original paper (which I got at the start of my project) the extension of the encoding for full π_{a}^{V} was done using something called "semantic substitution", which I will not detail here as I didn't have time to work on it.

If it works (and we hope it does!) this would give an encoding from π_{a}^{V} to TyCO that is fully abstract with respect to weak bisimulation, which would mean that indeed having nested variants and separate `case` analysis does *not* provide additional expressive power.

A Simple Type Systems

The table 10 gives typing rules for π_{a}^{V} and TyCO terms. Note that the usual subtyping rules are expressed by the $k \in J$ in some rules.

Similarly to receptive typing, the rules for bound output and dynamic links can be derived from the others.

Common Part

$$\frac{-}{\Sigma \vdash_{\mathrm{S}} \mathbf{0}} \; (\text{S-Nil}) \qquad \frac{-}{\Sigma, a{:}T \vdash_{\mathrm{S}} a{:}T} \; (\text{S-Nam})$$

$$\frac{\Sigma, x{:}T_1 \vdash_{\mathrm{S}} P \quad T_1 = T_2}{\Sigma, x{:}T_2 \vdash_{\mathrm{S}} P} \; (\text{S-Equ}_1) \qquad \frac{\Sigma \vdash_{\mathrm{S}} v{:}T_1 \quad T_1 = T_2}{\Sigma \vdash_{\mathrm{S}} v{:}T_2} \; (\text{S-Equ}_2)$$

$$\frac{\Sigma, x{:}T \vdash_{\mathrm{S}} P}{\Sigma \vdash_{\mathrm{S}} (\boldsymbol{\nu} x)\, P} \; (\text{S-Res}) \qquad \frac{\Sigma \vdash_{\mathrm{S}} P_1 \quad \Sigma \vdash_{\mathrm{S}} P_2}{\Sigma \vdash_{\mathrm{S}} P_1 | P_2} \; (\text{S-Par})$$

π_{a}^{V}

$$\frac{\Sigma \vdash_{\mathrm{S}} a{:}[T] \qquad \Sigma \vdash_{\mathrm{S}} v{:}T}{\Sigma \vdash_{\mathrm{S}} a!v} \; (\text{SP-Out})$$

$$\frac{\Sigma \vdash_{\mathrm{S}} a{:}[T] \qquad \Sigma, x{:}T \vdash_{\mathrm{S}} P}{\Sigma \vdash_{\mathrm{S}} a?(x).P} \; (\text{SP-Inp})$$

$$\frac{\Sigma \vdash_{\mathrm{S}} a{:}[T] \qquad \Sigma, x{:}T \vdash_{\mathrm{S}} P}{\Sigma \vdash_{\mathrm{S}} a?^{*}(x)P} \; (\text{SP-Rep})$$

$$\frac{(\forall j{\in}J)\;\; \Sigma, x_j{:}T_j \vdash_{\mathrm{S}} P_j \qquad \Sigma \vdash_{\mathrm{S}} v : \{\mathrm{l}_j{:}T_j \mid j{\in}J\}}{\Sigma \vdash_{\mathrm{S}} \mathsf{case}\, v\, \mathsf{of}\, \{\mathrm{l}_j(x_j){=}P_j \mid j{\in}J\}} \; (\text{SP-Case})$$

$$\frac{k{\in}J \qquad \Sigma \vdash_{\mathrm{S}} v{:}T_k}{\Sigma \vdash_{\mathrm{S}} \mathrm{l}_k\langle v\rangle : \{\mathrm{l}_j{:}T_j \mid j{\in}J\}} \; (\text{SP-Vval})$$

TyCO

$$\frac{\Sigma \vdash_{\mathrm{S}} a : [\{\mathrm{l}_j{:}T_j \mid j{\in}J\}] \qquad k{\in}J \qquad \Sigma \vdash_{\mathrm{S}} b{:}T_k}{\Sigma \vdash_{\mathrm{S}} a!\mathrm{l}_k\langle b\rangle} \; (\text{ST-Vout})$$

$$\frac{\Sigma \vdash_{\mathrm{S}} a : [\{\mathrm{l}_j{:}T_j \mid j{\in}J\}] \qquad (\forall j{\in}J)\;\; \Sigma, x_j{:}T_j \vdash_{\mathrm{S}} P_j}{\Sigma \vdash_{\mathrm{S}} a?\{\mathrm{l}_j(x_j){=}P_j \mid j{\in}J\}} \; (\text{ST-Vinp})$$

$$\frac{\Sigma \vdash_{\mathrm{S}} a : [\{\mathrm{l}_j{:}T_j \mid j{\in}J\}] \qquad (\forall j{\in}J)\;\; \Sigma, x_j{:}T_j \vdash_{\mathrm{S}} P_j}{\Sigma \vdash_{\mathrm{S}} a?^{*}\{\mathrm{l}_j(x_j){=}P_j \mid j{\in}J\}} \; (\text{ST-Rep})$$

Table 10: Simple Types for π_{a}^{V} and TyCO

Lemma A.0.1 (Substitution). *Let P be a π_{a}^{V}-term with $\Sigma \vdash_{\mathrm{S}} P$. If $\Sigma\{^{v}\!/_{x}\}$ is defined (i.e., type-correct), then $\Sigma\{^{v}\!/_{x}\} \vdash_{\mathrm{S}} P\{^{v}\!/_{x}\}$.*

The following lemmas hold for both π_{a}^{V}- and TyCO-terms.

Lemma A.0.2 (Typability of sub-terms). *If $\Sigma \vdash_{\mathrm{S}} P$ and Q is a sub-term of P then $\exists \Sigma'\ \Sigma' \vdash_{\mathrm{S}} Q$.*

Proposition A.0.3 (Subject Reduction in π_{a}^{V} and TyCO). *Let $\Sigma \vdash_{\mathrm{S}} P$.*

1. *If $P \rightarrow P'$, then $\Sigma \vdash_{\mathrm{S}} P'$.*
2. *If $P \xrightarrow{a!v} P'$, then $\Sigma \vdash_{\mathrm{S}} P'$.*
3. *If $P \xrightarrow{(\boldsymbol{\nu} b)\, a!v} P'$, then $\Sigma, b{:}T \vdash_{\mathrm{S}} P'$ for some T.*
4. *Let P be a π_{a}^{V} process and Q a TyCO process.*
   (a) *If $P \xrightarrow{a?v} P'$ and $\Sigma, a{:}[T] \vdash_{\mathrm{S}} v{:}T$, then $\Sigma, b_v{:}T' \vdash_{\mathrm{S}} P'$ for some T'.*
   (b) *If $Q \xrightarrow{a?\mathrm{l}\langle b\rangle} Q'$ and $\Sigma, a : [\{\mathrm{l}_j{:}T_j \mid j{\in}J\}] \vdash_{\mathrm{S}} b{:}T$, where $\mathrm{l} = \mathrm{l}_j$ and $T = T_j$ for some $j{\in}J$, then $\Sigma, b{:}T \vdash_{\mathrm{S}} P'$*

The notation b_v is a shorthand for extracting *the* name—there is exactly one—sitting inside v.

$$\frac{\Sigma \vdash_{\mathrm{S}} a{:}[\{\mathrm{l}_j{:}T_j \mid j{\in}J\}] \qquad k{\in}J \qquad \Sigma, b{:}T_k \vdash_{\mathrm{S}} P}{\Sigma \vdash_{\mathrm{S}} a!\mathrm{l}_k(\boldsymbol{\nu} b).P}\ (\text{ST-Bout})$$

$$\frac{\Sigma \vdash_{\mathrm{S}} a, b : [T]}{\Sigma \vdash_{\mathrm{S}} a \gg b}\ (\text{ST-Link})$$

Table 11: Simple TyCO Types — Derived Rules

B Proofs

B.1 TyCO to π_{a}^{V} Operational Correspondence (Proposition 3.0.1)

Lemma B.1.1 (Free Name and Substitution Homomorphism). *Let P be a TyCO process. Then :*

- $\mathrm{fn}(P) = \mathrm{fn}(\llbracket P \rrbracket)$
- $\llbracket P \rrbracket\{^a\!/_b\} = \llbracket P\{^a\!/_b\} \rrbracket$

Proof. By induction on the structure of P (the encoding only introduces fresh bound names for translation of input expressions) $\square$

For shortness we will write $P \overset{(\mu)}{\rightarrowtail} P'$ meaning the following :

1. If μ is τ or an input, then $P \rightarrowtail P'$
2. Otherwise $P = P'$.

First the proof of the first part of the Proposition :

We want to prove that $P \xrightarrow{\mu} P'$ implies $\llbracket P \rrbracket \xrightarrow{\mu} \overset{(\mu)}{\rightarrowtail} \llbracket P' \rrbracket$, for all P.

For each rule of the (TyCO) operational semantics, we assume we have a particular instance that match the premises and any side conditions, and (induction hypothesis) that the Proposition is true in the transitions (if any) in the premises of this instance.

We then prove that the Proposition is true on the transition that instance generates.

- (Alpha) : Assume $P \equiv_\alpha P' \xrightarrow{\mu} P''$ and $\llbracket P' \rrbracket \xrightarrow{\mu} \widetilde{P''} \overset{(\mu)}{\rightarrowtail} \llbracket P'' \rrbracket$ (where $\equiv_\alpha$ stands for α-equivalence).
  We have ($\equiv$-full abstraction, of which the proof is in the next subsection and does not use this Proposition), $P \equiv_\alpha P'$ implies $\llbracket P \rrbracket \equiv_\alpha \llbracket P' \rrbracket$.
  So, applying (Alpha), $P \xrightarrow{\mu} P''$ and $\llbracket P \rrbracket \xrightarrow{\mu} \widetilde{P''} \overset{(\mu)}{\rightarrowtail} \llbracket P'' \rrbracket$.
- (Out) : This is a base case as there are no transition in the premises in the rule.
  For any transition $c!\mathtt{l}\langle x\rangle \xrightarrow{c!\mathtt{l}\langle x\rangle} \mathbf{0}$ it generates we can also get the corresponding π_{a}^V one : $\llbracket c!\mathtt{l}\langle x\rangle \rrbracket = c!\mathtt{l}\langle x\rangle \xrightarrow{c!\mathtt{l}\langle x\rangle} \mathbf{0} = \llbracket \mathbf{0} \rrbracket$
- (Open) : Let $P \xrightarrow{c!\mathtt{l}\langle x\rangle} P'$ with $c \neq x$ (required to apply the rule) and (by induction hypothesis) $\llbracket P \rrbracket \xrightarrow{c!\mathtt{l}\langle x\rangle} \llbracket P' \rrbracket$.
  The transition provided by (Open) is
  $(\boldsymbol{\nu} x)\, P \xrightarrow{(\boldsymbol{\nu} x)\, c!\mathtt{l}\langle x\rangle} P'$ and
  $\llbracket (\boldsymbol{\nu} x)\, P \rrbracket = (\boldsymbol{\nu} x)\, \llbracket P \rrbracket$ (distributed encoding)
  $\xrightarrow{(\boldsymbol{\nu} x)\, c!\mathtt{l}\langle x\rangle} \llbracket P' \rrbracket$ ((Open), the side condition still applies because $c \neq x$)

- (Com$_1$) : Let $P_1 \xrightarrow{(\boldsymbol{\nu} x)\, c!\mathrm{l}\langle x\rangle} P_1'$, $P_2 \xrightarrow{c?\mathrm{l}\langle x\rangle} P_2'$ with $x \notin \mathrm{fn}(P_2)$.
By induction hypothesis, $[\![\, P_1 \,]\!] \xrightarrow{(\boldsymbol{\nu} x)\, c!\mathrm{l}\langle x\rangle} [\![\, P_1' \,]\!]$ and $[\![\, P_2 \,]\!] \xrightarrow{c?\mathrm{l}\langle x\rangle} \widetilde{P}_2' \rightarrowtail [\![\, P_2' \,]\!]$.
The transition produced by the rule is $P_1 \mid P_2 \xrightarrow{\tau} (\boldsymbol{\nu} x)\, P_1' \mid P_2'$.
Then, $[\![\, P_1 \mid P_2 \,]\!] = [\![\, P_1 \,]\!] \mid [\![\, P_2 \,]\!] \xrightarrow{\tau} (\boldsymbol{\nu} x)\, [\![\, P_1' \,]\!] \mid \widetilde{P}_2'$. (The side condition is true because $x \notin \mathrm{fn}(P_2) = \mathrm{fn}([\![\, P_2 \,]\!])$)
$\rightarrowtail (\boldsymbol{\nu} x)\, [\![\, P_1' \,]\!] \mid [\![\, P_2' \,]\!] = [\![\, (\boldsymbol{\nu} x)\, P_1' \mid P_2' \,]\!]$.
- (Par$_1$) : Let $P_1 \xrightarrow{\mu} P_1'$ with $\mathrm{bn}(\mu) \cap \mathrm{fn}(P_2) = \emptyset$ and (induction hypothesis) $[\![\, P_1 \,]\!] \xrightarrow{\mu} \widetilde{P}_1' \overset{(\mu)}{\rightarrowtail} [\![\, P_1' \,]\!]$.
Then: $P_1 \mid P_2 \xrightarrow{\mu} P_1' \mid P_2$ and (because $\mathrm{fn}(P_2) = \mathrm{fn}([\![\, P_2 \,]\!])$, the side condition is still true in the π_{a}^{V} side) $[\![\, P_1 \mid P_2 \,]\!] = [\![\, P_1 \,]\!] \mid [\![\, P_2 \,]\!] \xrightarrow{\mu} \widetilde{P}_1' \mid [\![\, P_2 \,]\!] \overset{(\mu)}{\rightarrowtail} [\![\, P_1' \,]\!] \mid [\![\, P_2 \,]\!] = [\![\, P_1' \mid P_2 \,]\!]$.
- (Res) : Let $P \xrightarrow{\mu} P'$ and (by induction) $[\![\, P \,]\!] \xrightarrow{\mu} \widetilde{P}' \overset{(\mu)}{\rightarrowtail} [\![\, P' \,]\!]$, with $a \notin \mathrm{n}(\mu)$.
The rule yields $(\boldsymbol{\nu} a)\, P \xrightarrow{\mu} (\boldsymbol{\nu} a)\, P'$ and $[\![\, (\boldsymbol{\nu} a)\, P \,]\!] = (\boldsymbol{\nu} a)\, [\![\, P \,]\!] \xrightarrow{\mu} (\boldsymbol{\nu} a)\, \widetilde{P}' \overset{(\mu)}{\rightarrowtail} (\boldsymbol{\nu} a)\, [\![\, P' \,]\!] = [\![\, (\boldsymbol{\nu} a)\, P' \,]\!]$. (the side condition $a \notin \mathrm{n}(\mu)$ is the same on both the TyCO and π_{a}^{V} side)
- (Inp) : (This is a base case as there are no transitions in the premises)
Let $k \in J$. We then have $c?\{\mathrm{l}_j(x_j){=}P_j \mid j{\in}J\} \xrightarrow{c?\mathrm{l}_k\langle v\rangle} P_k\{{}^{v}\!/\!_{x_k}\}$, and
$[\![\, c?\{\mathrm{l}_j(x_j){=}P_j \mid j{\in}J\} \,]\!] = c?(u).\mathsf{case}\, u\, \mathsf{of}\, \{\mathrm{l}_j(x_j){=}[\![\, P_j \,]\!] \mid j{\in}J\}$
$\xrightarrow{c?\mathrm{l}_k\langle v\rangle} \mathsf{case}\, \mathrm{l}_k\langle v\rangle\, \mathsf{of}\, \{\mathrm{l}_j(x_j){=}[\![\, P_j \,]\!] \mid j{\in}J\} \rightarrowtail [\![\, P_k \,]\!]\{{}^{v}\!/\!_{x_k}\}$
$= [\![\, P_k\{{}^{v}\!/\!_{x_k}\} \,]\!]$
- (Rep) : This is similar to (Inp) (and is a base case also). Let $k \in J$.
We then have $c?^*\{\mathrm{l}_j(x_j){=}P_j \mid j{\in}J\} \xrightarrow{c?\mathrm{l}_k\langle v\rangle}$
$(P_k\{{}^{v}\!/\!_{x_k}\}) \mid c?^*\{\mathrm{l}_j(x_j){=}P_j \mid j{\in}J\}$, and $[\![\, c?^*\{\mathrm{l}_j(x_j){=}P_j \mid j{\in}J\} \,]\!] =$
$c?^*(u).\mathsf{case}\, u\, \mathsf{of}\, \{\mathrm{l}_j(x_j){=}[\![\, P_j \,]\!] \mid j{\in}J\} \xrightarrow{c?\mathrm{l}_k\langle v\rangle}$

$$\begin{aligned}&\mathsf{case}\, \mathrm{l}_k\langle v\rangle\, \mathsf{of}\, \{\mathrm{l}_j(x_j){=}[\![\, P_j \,]\!] \mid j{\in}J\} \mid \\ &\qquad\qquad c?^*(u).\mathsf{case}\, u\, \mathsf{of}\, \{\mathrm{l}_j(x_j){=}[\![\, P_j \,]\!] \mid j{\in}J\} \rightarrowtail\end{aligned}$$

$[\![\, P_k \,]\!]\{{}^{v}\!/\!_{x_k}\} \mid c?^*(u).\mathsf{case}\, u\, \mathsf{of}\, \{\mathrm{l}_j(x_j){=}[\![\, P_j \,]\!] \mid j{\in}J\} =$
$[\![\, P_k\{{}^{v}\!/\!_{x_k}\} \,]\!] \mid [\![\, c?^*\{\mathrm{l}_j(x_j){=}P_j \mid j{\in}J\} \,]\!] =$
$[\![\, P_k\{{}^{v}\!/\!_{x_k}\} \mid c?^*\{\mathrm{l}_j(x_j){=}P_j \mid j{\in}J\} \,]\!]$.

Using the notation introduced above, the second part of the proposition can be expressed like this :

If $[\![P]\!] \xrightarrow{\mu} Q$ then there is P' such that $P \xrightarrow{\mu} P'$ and $Q \overset{(\mu)}{\rightarrowtail} [\![P']\!]$.

The second part of the proposition is proven in a similar way but working on the π_{a}^{V} operational semantics. Note however that, in the rules (Inp) and (Rep) we require the input actions to input a labeled value because there is no $\xrightarrow{c?u}$ transition in TyCO.

- (Alpha) : Let $\widetilde{P} \equiv_{\alpha} [\![P']\!] \xrightarrow{\mu} Q$, and (for some P'', induction hypothesis) $P' \xrightarrow{\mu} P''$ with $Q \overset{(\mu)}{\rightarrowtail} [\![P'']\!]$.
The generated transition is $\widetilde{P} \xrightarrow{\mu} Q$. Because we are only interested in transitions from an encoded term (otherwise the Proposition doesn't say anything) we will additionally assume that there is P s.t. $\widetilde{P} = [\![P]\!]$. So $[\![P]\!] \xrightarrow{\mu} Q$ is the resulting transition.
Because the encoding is $\equiv$-fully abstract, $[\![P]\!] \equiv_{\alpha} [\![P']\!]$ implies $P \equiv_{\alpha} P'$. So, by (Alpha), $P \xrightarrow{\mu} P''$. We already had $Q \overset{(\mu)}{\rightarrowtail} [\![P'']\!]$ by hypothesis.
- (Out) : (Base case) $[\![c!\mathrm{l}\langle x\rangle]\!] = c!\mathrm{l}\langle x\rangle \xrightarrow{c!\mathrm{l}\langle x\rangle} \mathbf{0} = [\![\mathbf{0}]\!]$ and $c!\mathrm{l}\langle x\rangle \xrightarrow{c!\mathrm{l}\langle x\rangle} \mathbf{0}$.
- (Open) : Let $[\![P]\!] \xrightarrow{c!\mathrm{l}\langle x\rangle} \widetilde{P}'$ with $c \neq x$ (required for applying (Open)) and (induction hypothesis) $P \xrightarrow{c!\mathrm{l}\langle x\rangle} P'$, $\widetilde{P}' = [\![P']\!]$.
(Open) yields (the side condition is fulfilled by $c \neq x$) $[\![(\boldsymbol{\nu} x)\, P]\!] = (\boldsymbol{\nu} x)\, [\![P]\!] \xrightarrow{(\boldsymbol{\nu} x)\, c!\mathrm{l}\langle x\rangle} \widetilde{P}'$ and (also applying (Open) - the side condition is the same) $(\boldsymbol{\nu} x)\, P \xrightarrow{(\boldsymbol{\nu} x)\, c!\mathrm{l}\langle x\rangle} P'$.
- (Com$_1$) :
Let $[\![P_1]\!] \xrightarrow{(\boldsymbol{\nu} x)\, c!\mathrm{l}\langle x\rangle} \widetilde{P}_1'$ and $[\![P_2]\!] \xrightarrow{c?\mathrm{l}\langle x\rangle} \widetilde{P}_2'$ with $x \notin \mathrm{fn}([\![P_2]\!])$ (which is required for applying the rule).
By induction hypothesis $P_1 \xrightarrow{(\boldsymbol{\nu} x)\, c!\mathrm{l}\langle x\rangle} P_1'$, $P_2 \xrightarrow{c?\mathrm{l}\langle x\rangle} P_2'$ and $\widetilde{P}_1' = [\![P_1']\!]$, $\widetilde{P}_2' \rightarrowtail [\![P_2']\!]$
The rule yields $[\![P_1 \mid P_2]\!] = [\![P_1]\!] \mid [\![P_2]\!] \xrightarrow{\tau} (\boldsymbol{\nu} x)\, \widetilde{P}_1' \mid \widetilde{P}_2'$ which $\rightarrowtail (\boldsymbol{\nu} x)\, [\![P_1']\!] \mid [\![P_2']\!] = [\![(\boldsymbol{\nu} x)\, P_1' \mid P_2']\!]$ and $P_1 \mid P_2 \xrightarrow{\tau} (\boldsymbol{\nu} x)\, P_1' \mid P_2'$. The side condition holds in both case because $\mathrm{fn}([\![P_2]\!]) = \mathrm{fn}(P_2)$.
- (Par$_1$) : Let $[\![P_1]\!] \xrightarrow{\mu} \widetilde{P}_1'$ with $\mathrm{bn}(\mu) \cap \mathrm{fn}(P_2) = \emptyset$.

By induction hypothesis, $\widetilde{P}'_1 \overset{(\mu)}{\rightarrowtail} [\![P'_1]\!]$ and $P_1 \xrightarrow{\mu} P'_1$.
Then: $[\![P_1 \mid P_2]\!] = [\![P_1]\!] \mid [\![P_2]\!] \xrightarrow{\mu} \widetilde{P}'_1 \mid [\![P_2]\!] \overset{(\mu)}{\rightarrowtail} [\![P'_1]\!] \mid [\![P_2]\!] = [\![P'_1 \mid P_2]\!]$ and $P_1 \mid P_2 \xrightarrow{\mu} P'_1 \mid P_2$.

- (Res) : Let $a \notin \mathrm{n}(\mu)$ and $[\![P]\!] \xrightarrow{\mu} \widetilde{P}'$ and (i. h.) $\rightarrowtail [\![P']\!]$ and $P \xrightarrow{\mu} P'$.
  (Res) yields $[\![(\boldsymbol{\nu} a)\, P]\!] = (\boldsymbol{\nu} a)\, [\![P]\!] \xrightarrow{\mu} (\boldsymbol{\nu} a)\, \widetilde{P}' \overset{(\mu)}{\rightarrowtail} (\boldsymbol{\nu} a)\, [\![P']\!]$
  $= [\![(\boldsymbol{\nu} a)\, P']\!]$ and $(\boldsymbol{\nu} a)\, P \xrightarrow{\mu} (\boldsymbol{\nu} a)\, P'$.
- (Inp) : This is a base case as there are no transitions in the premises. A transition $[\![P]\!] \xrightarrow{\mu} \widetilde{P}'$ produced by this rule has to be of the form
  $c?(u).\mathsf{case}\, u\, \mathsf{of}\, \{\mathrm{l}_j(x_j){=}[\![P_j]\!] \mid j{\in}J\} \xrightarrow{c?\mathrm{l_k}\langle x\rangle}$
  $\mathsf{case}\, \mathrm{l_k}\langle x\rangle\, \mathsf{of}\, \{\mathrm{l}_j(x_j){=}[\![P_j]\!] \mid j{\in}J\} = \widetilde{P}'$
  (so $P = c?\{\mathrm{l}_j(x_j){=}P_j \mid j{\in}J\}$).
  Then
  $\widetilde{P}' \rightarrowtail [\![P_k]\!]\{^x/_{x_k}\} = [\![P_k\{^x/_{x_k}\}]\!]$ and
  $P = c?\{\mathrm{l}_j(x_j){=}P_j \mid j{\in}J\} \xrightarrow{c?\mathrm{l_k}\langle x\rangle} P_k\{^x/_{x_k}\}$ (using TyCO's (Inp)).
- (Rep) :
  This one is similar to (Inp), just keeping the original process P in parallel.
- (P-Case) :
  All transitions produced by this rule are on a process that is a case statement and we can easily check that the encoding never produces such a statement (it does, but only prefixed by an input which is not the case here) which means that the proposition does not apply in this case.

B.2 Case Reduction "Transparency" (Lemma 2.2.5)

1. The first case occurs if the μ transition involves a subprocess of the expression P_k that was contained in the case expression that was reduced in $P \rightarrowtail P_0$. ("involves" means here that the proof of $P_0 \xrightarrow{\mu} P'_0$ applies (Out), (Inp) or (Rep) to it). In that case, rule (P-Case) can be applied on that expression, and have the case expression itself have the same transition as that of P_k. Inserting that step in the proof of $P_0 \xrightarrow{\mu} P'_0$ gives the proof of $P \xrightarrow{\mu} P'_0$.
   The second case occurs if that case expression is not involved in the

proof of $P_0 \xrightarrow{\mu} P'_0$ ("not involved" means that it only appears as P_2 in some applications of (Par$_1$)), then the proof of $P \xrightarrow{\mu} P'$ works exactly the same way (possibly using (Alpha) to avoid name collisions in other branches of the case expression).
Because the case expression was preserved, we then have $P' \rightarrowtail P'_0$.

2. If the proof of $P \xrightarrow{\mu} P'$ required (P-Case) doing the $P \rightarrowtail P_0$ reduction then $P_0 \xrightarrow{\mu} P'$ was actually a part of that proof.
Otherwise, if $P \xrightarrow{\mu} P'$ doesn't reduce that case expression then it can be still be reduced in P' (so we get $P' \rightarrowtail P'_0$) and the proof of $P \xrightarrow{\mu} P'$ can be directly transposed to $P_0 \xrightarrow{\mu} P'_0$ (and in this case the sets of free names will only shrink so no (Alpha) will be needed)

B.3 TyCO to π^V_{a} Embedding bisimilarity (Proposition 3.0.2)

First a proof of the "only if" part.
In the following, $\rightarrowtail^*$ is the transitive reflexive closure of $\rightarrowtail$.
Let $\mathcal{R}$ relate TyCO and π^V_{a} processes (P, Q) s.t. $Q \rightarrowtail^* [\![P]\!]$.
We will prove that the symmetric closure of $\mathcal{R}$ is a strong bisimulation.

- Let $P \mathrel{\mathcal{R}} Q$ and suppose $P \xrightarrow{\mu} P'$.
By 3.0.1, $[\![P]\!] \xrightarrow{\mu} \widetilde{P'} \overset{(\mu)}{\rightarrowtail} [\![P']\!]$.
Repeatedly applying 2.2.5 over $Q \rightarrowtail^* [\![P]\!] \xrightarrow{\mu} \widetilde{P'}$, we get $Q \xrightarrow{\mu} Q' \rightarrowtail^* \widetilde{P'} \overset{(\mu)}{\rightarrowtail} [\![P']\!]$ so $P' \mathrel{\mathcal{R}} Q'$.
- Now, let $P \mathrel{\mathcal{R}} Q$ and $Q \xrightarrow{\mu} Q'$.
Repeatedly applying 2.2.5 on $Q \rightarrowtail^* [\![P]\!]$, $[\![P]\!] \xrightarrow{\mu} Q''$ and $Q' \rightarrowtail^* Q''$.
By 3.0.1, $P \xrightarrow{\mu} P'$ and $Q'' \overset{(\mu)}{\rightarrowtail} [\![P']\!]$ so $Q' \rightarrowtail^* [\![P']\!]$ and $P' \mathrel{\mathcal{R}} Q'$.

B.4 TyCO to π^V_{a} $\equiv$-Full Abstraction (Prop. 3.0.3)

Let's first introduce a restricted structural congruence relation :
$A \equiv_\omega B$ is true if we have $A \equiv B$ without using α-renaming.
The encoding has three properties that will help for the proof.

1. For any (TyCO) context $C[\cdot]$ there is a (π^V_{a}) context $C'[\cdot]$ such that

$\forall P : [\![C[P]]\!] = C'[[\![P]\!]]$ and
$\forall P, Q : [\![P]\!] = C'[Q]$ implies $\exists P'([\![P']\!] = Q)$

2. It doesn't create parallel composition or restriction operators, which, together with the fact that it is a distributed encoding, implies that $[\![P]\!] = A|B$ implies $\exists! A', B'(P = A'|B')$ (so $[\![A']\!] = A$ and $[\![B']\!] = B$), and $[\![P]\!] = (\boldsymbol{\nu} a)\, A$ implies $\exists! A'(P = (\boldsymbol{\nu} a)\, A')$ (so $[\![A']\!] = A$).
3. It is injective, ie $[\![P]\!] = [\![Q]\!]$ implies $P = Q$.

Translating $C[\cdot]$ into $C'[\cdot]$ for point 1 is done with the same rules as the encoding and adding a rule translating a hole into a hole.

Then checking that each rule of the encoding respects point 2 is easy.

To prove the last property we can just see that if two processes differ then the encoding will produce different processes as well :

If they are using a different syntactic construct then those produced by the encoding will be different. If they are using the same syntactic construct but with different names then the names will differ accordingly on the π^V_{a} side.

Alpha-convertibility is preserved because $\mathrm{fn}([\![P]\!]) = \mathrm{fn}(P)$.

The last two properties imply that $\forall P, Q, P \equiv_\omega Q$ implies $[\![P]\!] \equiv_\omega [\![Q]\!]$, and $\forall P, Q, [\![P]\!] \equiv_\omega Q$ implies $\exists Q'(P \equiv_\omega Q'$ and $[\![Q']\!] \equiv_\alpha Q)$. (We don't have equality because of the u name introduced in the translation of input expressions)

As an example, here's the proof for the case $X|Y \equiv_\omega Y|X$.

$[\![P]\!] = [\![X]\!] | [\![Y]\!] \equiv [\![Y]\!] | [\![X]\!] = [\![Y|X]\!] = [\![Q]\!]$.

On the other direction :

$[\![P]\!] = A|B \equiv_\omega B|A = Q$. Then there are A' and B' such that $P = A'|B'$ and $[\![A']\!] = A$ and $[\![B']\!] = B$ so we have $Q' = B'|A'$ and we clearly see $P \equiv_\omega Q'$.

Property number one extends the proof to the closure as a congruence.

B.5 R-Bout (Table 7)

We will see what are the conditions for a typing of $a!\mathrm{l}(\boldsymbol{\nu} b).P_b$, $(\boldsymbol{\nu} b)\,(a!\mathrm{l}\langle b\rangle | P_b)$.

Both input and output rules allow weakening on linear names but we will do it on the output side only because it is simpler and leads to the same result (so we instantiate Φ to $\emptyset$ in the input rules).

This will actually lead to a rule that is stricter than necessary but that will look consistent with the other rules. However it just means that weakening that could be done with (Inp) or (Rep) for P_b will have to be done at (R-Bout)-time and will yield to the same result.

- Let $\Delta_{b1}; \Gamma_{b1}; \Delta_{b\mathrm{u}}; \Gamma_{b\mathrm{u}} \vdash_{\mathrm{R}} P_b$.

Because P_b is an input at channel b, the input rules will require
$\Delta_{b1} = \{b\}_1$, $\Delta_{b\mathrm{u}} = \{b\}_{\mathrm{u}}$ and $b \notin \Gamma_{b1}$
For the left component of the parallel composition we have (assuming $\Phi_1 \cap \{a, b\}_1 = \emptyset$) :

- $\Phi_1; \Phi_1 \oplus \{a, b\}_1; \emptyset; \Gamma_{b\mathrm{u}} \vdash_{\mathrm{R}} a!\mathrm{l}\langle b\rangle$

With condition that $\{a, b\}_{\mathrm{u}} \subseteq \Gamma_{b\mathrm{u}}$ and $\{a\}_1 \cap \{b\}_1 = \emptyset$.
$\Phi_1 \cap \{a, b\}_1 = \emptyset$ implies $b \notin \Phi_1$ so $\Delta_{b1} \cap \Phi_1 = \emptyset$.
To be able to apply the (R-Par) rule we also need
$\Gamma_{b1} \cap (\Phi_1 \cup \{a, b\}_1) = \emptyset$.
After parallel composition we get :

- $\{b\}_1 \oplus \Phi_1; \Gamma_{b1} \oplus \Phi_1 \oplus \{a, b\}_1; \{b\}_{\mathrm{u}}; \Gamma_{b\mathrm{u}} \vdash_{\mathrm{R}} a!\mathrm{l}\langle b\rangle \mid P_b$

The conditions of the (R-Res) are fulfilled (If b is receptive then the above judgment puts it into both Γ and Δ).
Therefore we obtain the following judgment for the complete process :
$\Phi_1; \Gamma_{b1} \oplus \Phi_1 \oplus \{a, b\}_1 \setminus \{b\}_1; \emptyset; \Gamma_{b\mathrm{u}} \setminus \{b\}_{\mathrm{u}} \vdash_{\mathrm{R}} a!\mathrm{l}(\boldsymbol{\nu} b).P_b$
which is equivalent to (thank to $\{a\}_1 \cap \{b\}_1 = \emptyset$)
$\Phi_1; \Gamma_{b1} \oplus \Phi_1 \oplus \{a\}_1; \emptyset; \Gamma_{b\mathrm{u}} \setminus \{b\}_{\mathrm{u}} \vdash_{\mathrm{R}} a!\mathrm{l}(\boldsymbol{\nu} b).P_b$
Defining $\Gamma_1 = \Gamma_{b1} \oplus \{a\}_1$ and $\Gamma_{\mathrm{u}} = \Gamma_{b\mathrm{u}} \setminus \{b\}_{\mathrm{u}}$, this now matches the (R-Bout) rule.

B.6 R-Link (Table 7)

Assume we have $\Delta_{01}; \Gamma_{01}; \Delta_{0\mathrm{u}}; \Gamma_{0\mathrm{u}} \vdash_{\mathrm{R}} z \gg x$.

We will then give the typing for $a \gg b$ which is defined to be either $a?^*\{\mathrm{l}_j(x) = b!\mathrm{l}_j(\boldsymbol{\nu} z).z \gg x \mid j \in J\}$ (for a non-linear) or $a?\{\mathrm{l}_j(x) = b!\mathrm{l}_j(\boldsymbol{\nu} z).z \gg x \mid j \in J\}$ for a linear. z is fresh.

With hypothesis $\Delta_{01} = \{z\}_1$, $\{b, z\}_1 \cap \Gamma_{01} = \emptyset$, $\Delta_{0\mathrm{u}} = \{z\}_{\mathrm{u}}$ and $\{b, z\}_{\mathrm{u}} \subseteq \Gamma_{0\mathrm{u}}$, we have (using (R-Bout)) :

$\emptyset; \Gamma_{01} \cup \{b\}_1; \emptyset; \Gamma_{0\mathrm{u}} \setminus \{z\}_{\mathrm{u}} \vdash_{\mathrm{R}} b!\mathrm{l}(\boldsymbol{\nu} z).z \gg x$.

Then, if a is non-linear, under hypothesis $b \notin \mathbf{N}_{\mathtt{l}}$, $\Gamma_{0\mathtt{l}} = \{x\}_{\mathtt{l}}$ and $\{x\}_{\mathtt{u}} \subseteq \Gamma_{0\mathtt{u}} \setminus \{z\}_{\mathtt{u}}$, we have (using (R-REP)) :

$\{a\}_{\mathtt{l}} \oplus \Phi_{\mathtt{l}}; \Phi_{\mathtt{l}}; \{a\}_{\mathtt{u}}; \Gamma_{0\mathtt{u}} \setminus \{x, z\}_{\mathtt{u}} \vdash_{\mathrm{R}} a \gg b$.

If a is linear, under hypothesis $a \notin \Gamma_{0\mathtt{l}} \cup \{b\}_{\mathtt{l}}$, $\{x\}_{\mathtt{l}} \subseteq \Gamma_{0\mathtt{l}} \cup \{b\}_{\mathtt{l}}$ and $\{x\}_{\mathtt{u}} \subseteq \Gamma_{0\mathtt{u}} \setminus \{z\}_{\mathtt{u}}$, we have (using (R-INP)) :

$\{a\}_{\mathtt{l}} \oplus \Phi_{\mathtt{l}}; (\Gamma_{0\mathtt{l}} \cup \{b\}_{\mathtt{l}}) \setminus \{x\}_{\mathtt{l}} \oplus \Phi_{\mathtt{l}}; \{a\}_{\mathtt{u}}; \Gamma_{0\mathtt{u}} \setminus \{x, z\}_{\mathtt{u}} \vdash_{\mathrm{R}} a \gg b$. Note that the hypothesis implies that if a and b are linear then they are different

Summing up all hypothesis we get the following :

The conditions for $\{a\}_{\mathtt{l}} \oplus \Phi_{\mathtt{l}}; \Gamma_{\mathtt{l}} \oplus \Phi_{\mathtt{l}}; \{a\}_{\mathtt{u}}; \Gamma_{\mathtt{u}} \vdash_{\mathrm{R}} a \gg b$ are :

$a \notin \mathbf{N}_{\mathtt{l}} \Rightarrow b \notin \mathbf{N}_{\mathtt{l}}$, $\{a, z\}_{\mathtt{l}} \cap \Gamma_{\mathtt{l}} = \emptyset$, $\{b\}_{\mathtt{l}} \subseteq \Gamma_{\mathtt{l}}$, $\{b\}_{\mathtt{u}} \subseteq \Gamma_{\mathtt{u}}$ and

$\{z\}_{\mathtt{l}}; \Gamma_{\mathtt{l}} \setminus \{b\}_{\mathtt{l}} \cup \{x\}_{\mathtt{l}}; \{z\}_{\mathtt{u}}; \Gamma_{\mathtt{u}} \cup \{x, z\}_{\mathtt{u}} \vdash_{\mathrm{R}} z \gg x$

We have $\{b\}_{\mathtt{l}} \subseteq \Gamma_{\mathtt{l}}$, and actually we require (as a rule) $\Gamma_{\mathtt{l}} = \{b\}_{\mathtt{l}}$ because it would not make sense to have other names in it : such a name would not be introduced by a rule ((R-BOUT) or (R-INP)) but just preserved from one recursion level to another. This undecidability arises from the fact that the definition of dynamic links is recursive and there may not be a base case that "initializes" $\Gamma_{\mathtt{l}}$ to $\emptyset$.

B.7 Weakening/Strengthening (Lemma 4.1.1)

In this proof we will give the exact conditions under which $\Theta \vdash_{\mathrm{R}} P$ is a valid judgment and we will get a proof of Lemma 4.1.3 for free.

We will prove that for all type-correct P there is one *strongest judgment* $\Delta_0; \Gamma_0 \vdash_{\mathrm{R}} P$ with $\Delta_0 \cup \Gamma_0 = \mathrm{fn}(P)_{\mathtt{r}}$ and a set of names $\widetilde{\Phi} \subseteq \mathrm{bn}(P)$ such that

$\Delta_{\mathtt{l}}; \Gamma_{\mathtt{l}}; \Delta_{\mathtt{u}}; \Gamma_{\mathtt{u}} \vdash_{\mathrm{R}} P$ iff, for some Φ :

- $\Phi \cap \widetilde{\Phi} = \emptyset$
- $\Delta_{\mathtt{l}} = \Delta_{0\mathtt{l}} \oplus \Phi_{\mathtt{l}}$
- $\Gamma_{\mathtt{l}} = \Gamma_{0\mathtt{l}} \oplus \Phi_{\mathtt{l}}$
- $\Delta_{0\mathtt{u}} = \Delta_{\mathtt{u}}$
- $\Gamma_{\mathtt{u}} = \Gamma_{0\mathtt{u}} \oplus \Phi_{\mathtt{u}}$

The reason why $\widetilde{\Phi}$ is not always equal to $\mathrm{bn}(P)$ is that if the binding of a linear name is "deep enough" then that name can be put (linear weakening) in $\Delta_{\mathtt{l}}$ and $\Gamma_{\mathtt{l}}$. However, $\widetilde{\Phi}_{\mathtt{u}} = \mathrm{bn}(P)_{\mathtt{u}}$. This will be explained in more details below.

We will show that the lemma is true for the conclusion of all the receptive typing rules, under hypothesis that it is true in the premises.

- (R-Nil)
  The (R-Nil) rule allows a typing of $\mathbf{0}$ for any $\Delta_{\mathtt{l}} = \Gamma_{\mathtt{l}}$ pair and for any $\Gamma_{\mathtt{u}}$.
  So for $\mathbf{0}$ the strongest judgment is $\emptyset; \emptyset; \emptyset; \emptyset \vdash_{\mathrm{R}} \mathbf{0}$ and $\widetilde{\Phi} = \emptyset$.
- (R-Out)
  Let $P = a!\mathrm{l}\langle b\rangle$. We have $\Phi_{\mathtt{l}}; \{a, b\}_{\mathtt{l}} \oplus \Phi_{\mathtt{l}}; \Phi_{\mathtt{u}}; \{a, b\}_{\mathtt{u}} \oplus \Phi_{\mathtt{u}} \vdash_{\mathrm{R}} P$ for any Φ so the strongest judgment is $\emptyset; \{a, b\}_{\mathtt{l}}; \emptyset; \{a, b\}_{\mathtt{u}} \vdash_{\mathrm{R}} P$ and $\widetilde{\Phi} = \emptyset$.
  We have $\mathrm{fn}(P) = \{a, b\}$ so the free names requirement holds
- (R-Par)
  Let $P = P_1|P_2$. Let $\Delta_{i\mathtt{l}}; \Gamma_{i\mathtt{l}}; \Delta_{i\mathtt{u}}; \Gamma_{i\mathtt{u}} \vdash_{\mathrm{R}} P_i$ for $i \in \{1, 2\}$ be the strongest judgment for P_i, and let $\widetilde{\Phi}_i$ be the corresponding restriction set. Note that $\Gamma_{1\mathtt{u}}$ and $\Gamma_{2\mathtt{u}}$ can be different.
  A valid judgment for P_i will then be as follows :
  For both $i = 1, 2$, Φ_i is chosen not intersecting $\widetilde{\Phi}_i$.
  $\Delta_{i\mathtt{l}} \oplus \Phi_{i\mathtt{l}}; \Gamma_{i\mathtt{l}} \oplus \Phi_{i\mathtt{l}}; \Delta_{i\mathtt{u}}; \Gamma_{i\mathtt{u}} \oplus \Phi_{i\mathtt{u}} \vdash_{\mathrm{R}} P_i$.
  To be able to apply (R-Par) Φ_i must be chosen in a way that $\Gamma_{1\mathtt{u}} \oplus \Phi_{1\mathtt{u}} = \Gamma_{2\mathtt{u}} \oplus \Phi_{2\mathtt{u}}$ (let's call the resulting set $\Gamma_{\mathtt{u}}$)
  So there is Φ_0 such that
  $\Phi_{1\mathtt{u}} = \Gamma_{1\mathtt{u}} \setminus \Gamma_{2\mathtt{u}} \oplus \Phi_0$
  $\Phi_{2\mathtt{u}} = \Gamma_{2\mathtt{u}} \setminus \Gamma_{1\mathtt{u}} \oplus \Phi_0$
  We then have $\Gamma_{\mathtt{u}} = \Gamma_{1\mathtt{u}} \oplus \Gamma_{2\mathtt{u}} \oplus \Phi_0$ with Φ_0 not intersecting $\widetilde{\Phi}_{\mathtt{u}} = \widetilde{\Phi}_{1\mathtt{u}} \cup \widetilde{\Phi}_{2\mathtt{u}}$.
  Because (R-Par) requires linear sets to be disjoint instead of equal the only additional constraint on $\Phi_{i\mathtt{l}}$ is that they must be disjoint from each other.
  Define $\Phi = \Phi_{1\mathtt{l}} \oplus \Phi_{2\mathtt{l}} \oplus \Phi_0$.
  We then have $\Delta_{1\mathtt{l}} \oplus \Delta_{2\mathtt{l}} \oplus \Phi_{\mathtt{l}}; \Gamma_{1\mathtt{l}} \oplus \Gamma_{2\mathtt{l}} \oplus \Phi_{\mathtt{l}}; \Delta_{1\mathtt{u}} \oplus \Delta_{2\mathtt{u}}; \Gamma_{\mathtt{u}} \oplus \Phi_{\mathtt{u}} \vdash_{\mathrm{R}} P$.
  Setting $\Phi = \emptyset$ gives the strongest judgment.
  As $\mathrm{fn}(P) = \mathrm{fn}(P_1) \cup \mathrm{fn}(P_2)$ the free names requirement holds.
  Because both $\Phi_{1\mathtt{l}}$ and $\Phi_{2\mathtt{l}}$ can provide their own linear weakening we have $\widetilde{\Phi}_{\mathtt{l}} = \widetilde{\Phi}_{1\mathtt{l}} \cap \widetilde{\Phi}_{2\mathtt{l}}$. (and the uniform subset is the one already given above)
- (R-Res)
  Let $P = (\boldsymbol{\nu} a)\, P_0$. Let $\Delta^{\star}_{\mathtt{l}}; \Gamma^{\star}_{\mathtt{l}}; \Delta^{\star}_{\mathtt{u}}; \Gamma^{\star}_{\mathtt{u}} \vdash_{\mathrm{R}} P_0$ the strongest judgment for P_0 and $\widetilde{\Phi}^{\star}$ defined appropriately.
  Let then $\Delta_{\mathtt{l}} \oplus \Phi_{\mathtt{l}}; \Gamma_{\mathtt{l}} \oplus \Phi_{\mathtt{l}}; \Delta_{\mathtt{u}}; \Gamma_{\mathtt{u}} \oplus \Phi_{\mathtt{u}} \vdash_{\mathrm{R}} P_0$ be a valid judgment.

(R-Res) will then yield
$\Delta_{\mathtt{l}} \setminus \{a\}_{\mathtt{l}} \oplus \Phi_{\mathtt{l}}; \Gamma_{\mathtt{l}} \setminus \{a\}_{\mathtt{l}} \oplus \Phi_{\mathtt{l}}; \Delta_{\mathtt{u}} \setminus \{a\}_{\mathtt{u}}; \Gamma_{\mathtt{u}}\{a\}_{\mathtt{u}} \oplus \Phi_{\mathtt{u}} \vdash_{\mathrm{R}} P$.
So the strongest judgment of P is obtained setting Φ to $\emptyset$ and is
$\Delta_{\mathtt{l}} \setminus \{a\}_{\mathtt{l}}; \Gamma_{\mathtt{l}} \setminus \{a\}_{\mathtt{l}}; \Delta_{\mathtt{u}} \setminus \{a\}_{\mathtt{u}}; \Gamma_{\mathtt{u}} \setminus \{a\}_{\mathtt{u}} \vdash_{\mathrm{R}} P$.
We have $\mathrm{fn}(P) = \mathrm{fn}(P_0) \setminus \{a\} = (\Delta \setminus \{a\}) \cup (\Gamma \setminus \{a\})$, as required.
Because $\{a\}_{\mathtt{l}}$ was a subset of $\Delta_{\mathtt{l}}$ and $\Gamma_{\mathtt{l}}$, and $\{a\}_{\mathtt{u}}$ of $\Delta_{\mathtt{u}}$ and $\Gamma_{\mathtt{u}}$, a could not be in $\Phi_{\mathtt{l}}$ or $\Phi_{\mathtt{u}}$. Because it has been removed by (R-Res) this constraint must be enforced for P: $\widetilde{\Phi}$ is defined to $\widetilde{\Phi}^{\star} \cup \{a\}$.

- (R-Inp)
Let $\Theta \vdash_{\mathrm{R}} P = a?\{\mathrm{l}_j(x_j){=}P_j \mid j{\in}J\}$.
For each j, let $\emptyset; \Gamma^{\star}_{\mathtt{l}} \cup \{x_j\}_{\mathtt{l}}; \emptyset; \Gamma^{\star}_{j_{\mathtt{u}}} \vdash_{\mathrm{R}} P_j$ be the strongest judgment for P_j and $\widetilde{\Phi}_j$ the corresponding restriction set.
This $\Gamma^{\star}_{\mathtt{l}}$ must be the same for all, as requested by (R-Inp), because all have an empty $\Delta^{\star}_{\mathtt{l}}$ so no linear weakening can be used to have them share the same $\Gamma_{\mathtt{l}}$.
However each may have a different $\Gamma^{\star}_{j_{\mathtt{u}}}$ because uniform weakening can make them all equal :
$$\text{Let } \Gamma^{\star}_{\mathtt{u}} = \bigcup_{j \in J} \left(\Gamma^{\star}_{j_{\mathtt{u}}} \setminus \{x_j\}_{\mathtt{u}} \right)$$
We have to make the union because these were all strongest judgment so the $\Gamma^{\star}_{j_{\mathtt{u}}}$ can only be extended. We know we will not hit any $\widetilde{\Phi}_j$ because we had assumed that P was typable.
With Φ not intersecting the set of x_j, applying (R-Inp) we get :
$\{a\}_{\mathtt{l}} \oplus \Phi_{\mathtt{l}}; \Gamma^{\star}_{\mathtt{l}} \oplus \Phi_{\mathtt{l}}; \emptyset; \Gamma^{\star}_{\mathtt{u}} \vdash_{\mathrm{R}} P$
Taking $\Phi = \emptyset$ we get the strongest judgment for P. The only constraint on Φ being not to contain any x_j, the corresponding $\widetilde{\Phi}$ can be defined to $\{x_j \mid j \in J\}$.
Note that (R-Inp) allows to put in Φ a name bound in one or more P_j.
Concerning free names :
By induction, $\mathrm{fn}(P_j)_{\mathtt{r}} = \Gamma^{\star}_{\mathtt{l}} \cup \{x_j\}_{\mathtt{l}} \cup \Gamma^{\star}_{j_{\mathtt{u}}}$
So
$\mathrm{fn}(P)_{\mathtt{r}} = \{a\}_{\mathtt{r}} \cup \bigcup_{j \in J} \mathrm{fn}(P_j) \setminus x_j = \{a\}_{\mathtt{r}} \cup \Gamma^{\star}_{\mathtt{l}} \cup \Gamma^{\star}_{\mathtt{u}}$
which is what was required.
- (R-Rep)
This case works in a similar way to the previous one ...

B.8 Type Soundness (4.1.6)

All entries in this proposition are of the following form :

Based on some μ, if $\Delta; \Gamma \vdash_{\mathrm{R}} P$ and $P \xrightarrow{\mu} Q$ then there are Δ^0, Γ^0, Δ^1, Γ^1, Δ^-, Γ^-, Δ^+, Γ^+ such that
$\Delta \setminus \Delta^0; \Gamma \setminus \Gamma^0 \vdash_{\mathrm{R}} P$,
$\Delta^1 \subseteq \Delta$ and $\Gamma^1 \subseteq \Gamma$,
$(\Delta \setminus \Delta^0) \cup \Delta^+ \setminus \Delta^-; (\Gamma \setminus \Delta^0) \cup \Gamma^+ \setminus \Gamma^- \vdash_{\mathrm{R}} Q$.
And for each entry these sets are always defined such that

- $\Delta^0 \cup \Gamma^0 \subseteq \mathrm{bn}(\mu)$
- Either: $\Delta^1 \cup \Gamma^1 \subseteq \mathrm{n}(\mu)$
  Or: (only if $\mu = \tau$) $\Delta^1 = \Gamma^1 = \{a\}$ for some a
- $\Delta^- \subseteq \Delta^1$ and $\Delta^+ \subseteq \Delta^0$ (and the same for $\Gamma^{\cdots}$)

Knowing this we can show that all transition rules preserve the proposition :

We consider an instance (any side conditions being true) of each operational semantic rule, and assume that the full (type soundness) proposition is true for all transitions in the premises of that rule and then show that it is true for any transition produced by that rule.

Because the proposition only applies on typable processes and operational semantics only produce transitions of typable processes if those in the premises are themselves typable we will assume that for all transitions that we consider the source process is typable.

For shortness, When a set is empty it will not be mentioned.

- (Out)
  According to (R-Out), $\Phi_{\mathtt{l}}; \{a,b\}_{\mathtt{l}} \oplus \Phi_{\mathtt{l}}; \emptyset; \{a,b\}_{\mathtt{u}} \oplus \Phi_{\mathtt{u}} \vdash_{\mathrm{R}} a!\mathtt{l}\langle b\rangle$ $(a \neq b)$. We also have $\Phi_{\mathtt{l}}; \Phi_{\mathtt{l}}; \emptyset; \{a,b\}_{\mathtt{u}} \oplus \Phi_{\mathtt{u}} \vdash_{\mathrm{R}} \mathbf{0}$.
  So for $P = a!\mathtt{l}\langle b\rangle$, $\Gamma^1 = \{a,b\}_{\mathtt{r}}$ and $\Gamma^- = \{a,b\}_{\mathtt{l}}$ match.
- (Open)
  Let $P \xrightarrow{a!\mathtt{l}\langle b\rangle} Q$ $(a \neq b)$. By induction there is Θ such that
  $\Delta_{\mathtt{l}}; \Gamma_{\mathtt{l}}, \{a,b\}_{\mathtt{l}}; \Delta_{\mathtt{u}}; \Gamma_{\mathtt{u}}, \{a,b\}_{\mathtt{u}} \vdash_{\mathrm{R}} P$.
  $\Delta_{\mathtt{l}}; \Gamma_{\mathtt{l}}; \Delta_{\mathtt{u}}; \Gamma_{\mathtt{u}}, \{a,b\}_{\mathtt{u}} \vdash_{\mathrm{R}} Q$.
  For typing $(\boldsymbol{\nu} b)\, P$, rule (R-Res) requires the following :
  $\{b\}_{\mathtt{l}} \subseteq \Delta_{\mathtt{l}}$ and $\{b\}_{\mathtt{u}} \subseteq \Delta_{\mathtt{u}}$.
  And yields :

$\Delta_{\mathtt{1}} \setminus \{b\}_{\mathtt{1}}; \Gamma_{\mathtt{1}}, \{a\}_{\mathtt{1}}; \Delta_{\mathtt{u}} \setminus \{b\}_{\mathtt{u}}; \Gamma_{\mathtt{u}}, \{a\}_{\mathtt{u}} \vdash_{\mathrm{R}} (\boldsymbol{\nu} b)\, P$.
So the result is (for $(\boldsymbol{\nu} b)\, P \xrightarrow{(\boldsymbol{\nu} b)\, a!1\langle b\rangle} Q$) obtained from (R-Res) requirements and checking the differences between the typings of $(\boldsymbol{\nu} b)\, P$ and Q :
$\Delta^0 = \{b\}$, $\Gamma^0 = \{b\}$, $\Gamma^1 = \{a\}_{\mathtt{r}}$, $\Delta^+ = \{b\}_{\mathtt{r}}$, $\Gamma^-_{\mathtt{1}} = \{a\}_{\mathtt{1}}$ and $\Gamma^+_{\mathtt{u}} = \{b\}_{\mathtt{u}}$.

- (Com$_1$)
Let $P_1 \xrightarrow{(\boldsymbol{\nu} b)\, a!1\langle b\rangle} P_1'$ and $P_2 \xrightarrow{a?1\langle b\rangle} P_2'$.
By induction (on the two above transitions), and applying (R-Par), we have
$\Delta_{\mathtt{1}}, \{a\}_{\mathtt{1}}; \Gamma_{\mathtt{1}}, \{a\}_{\mathtt{1}}; \Delta_{\mathtt{u}}, \{a\}_{\mathtt{u}}; \Gamma_{\mathtt{u}}, \{a\}_{\mathtt{u}} \vdash_{\mathrm{R}} P_1 \mid P_2$ and
$\Delta_{\mathtt{1}}, \{b\}_{\mathtt{1}}; \Gamma_{\mathtt{1}}, \{b\}_{\mathtt{1}}; \Delta_{\mathtt{u}}, \{a, b\}_{\mathtt{u}}; \Gamma_{\mathtt{u}}, \{a, b\}_{\mathtt{u}} \vdash_{\mathrm{R}} P_1' \mid P_2'$.
We also know that b is not in Γ.
Applying (R-Res),
$\Delta_{\mathtt{1}}; \Gamma_{\mathtt{1}}; \Delta_{\mathtt{u}}, \{a\}_{\mathtt{u}}; \Gamma_{\mathtt{u}}, \{a\}_{\mathtt{u}} \vdash_{\mathrm{R}} (\boldsymbol{\nu} b)\, P_1' \mid P_2'$.
The transition returned by (Com$_1$) is $P = P_1 \mid P_2 \xrightarrow{\tau} (\boldsymbol{\nu} b)\, P_1' \mid P_2' = Q$.
So, comparing the typings of P and Q (and dropping the fact that b is not in Γ, because it doesn't change)
We have $\Delta^1_{\mathtt{1}} = \Gamma^1_{\mathtt{1}} = \Delta^-_{\mathtt{1}} = \Gamma^-_{\mathtt{1}} = \{a\}_{\mathtt{1}}$ and $\Delta^1_{\mathtt{u}} = \Gamma^1_{\mathtt{u}} = \{a\}_{\mathtt{u}}$ as stated in the entry of the Proposition.
- (Par$_1$)
Let $P \xrightarrow{\mu} P'$ with Γ^0, Δ^0, Γ^1, etc, defined accordingly.
Let $\Delta_{P\mathtt{1}}; \Gamma_{P\mathtt{1}}; \Delta_{P\mathtt{u}}; \Gamma_{P\mathtt{u}} \vdash_{\mathrm{R}} P$ and
$\Delta_{Q_{\mathtt{1}}}; \Gamma_{Q_{\mathtt{1}}}; \Delta_{Q_{\mathtt{u}}}; \Gamma_{Q_{\mathtt{u}}} \vdash_{\mathrm{R}} Q$ that fulfills the requirements of (R-Par) (i.e. with $\Gamma_{P\mathtt{u}} = \Gamma_{Q_{\mathtt{u}}}$ and the three other set pairs disjoints) and such that $\mathrm{bn}(\mu) \cap \mathrm{fn}(Q) = \emptyset$, as required by (Par$_1$).
We then have $\Delta_{\mathtt{1}}; \Gamma_{\mathtt{1}}; \Delta_{\mathtt{u}}; \Gamma_{\mathtt{u}} \vdash_{\mathrm{R}} P|Q$ and $P|Q \xrightarrow{\mu} P'|Q$, with $\Delta_{\mathtt{1}} = \Delta_{P\mathtt{1}} \oplus \Delta_{Q_{\mathtt{1}}}$ (and the same for the two next sets, and $\Gamma_{\mathtt{u}} = \Gamma_{P\mathtt{u}} = \Gamma_{Q_{\mathtt{u}}}$).
Because $\Delta_P \subseteq \Delta$ and $\Gamma_P \subseteq \Gamma$, the inclusion property of Δ^1 and Γ^1 are preserved for Δ and Γ.
We have $\Gamma^0 \cup \Delta^0 \subseteq \mathrm{bn}(\mu)$ so because of the side condition of (Par), $(\Gamma^0 \cup \Delta^0) \cap \mathrm{fn}(Q) = \emptyset$ as well, which (4.1.1) implies that if they are in Δ_Q and $\Gamma_{Q_{\mathtt{1}}}$ then they can be removed :
With $\Phi_{\mathtt{1}} = (\Gamma^0 \cup \Delta^0)$,
$\Delta_{Q_{\mathtt{1}}} \setminus \Phi_{\mathtt{1}}; \Gamma_{Q_{\mathtt{1}}} \setminus \Phi_{\mathtt{1}}; \Delta_{Q_{\mathtt{u}}}; \Gamma_{Q_{\mathtt{u}}} \vdash_{\mathrm{R}} Q$
(This is the $\Phi_{\mathtt{1}}$ in point number 3). So Γ^0 and Δ^0 are preserved for Δ

and Γ.
By induction we had $\Delta_P \setminus \Delta^- \cup \Delta^+; \Gamma_P \setminus \Gamma^- \cup \Gamma^+ \vdash_{\mathrm{R}} P'$. So by (R-Par) (with $P' = P'|Q$),
$\Delta_Q \oplus (\Delta_P \setminus \Delta^- \cup \Delta^+); \Gamma_Q \cup (\Gamma_P \setminus \Gamma^- \cup \Gamma^+) \vdash_{\mathrm{R}} P'$. (as usual, Γ are disjoint in linear names and equal in uniform names).
Because $\Delta^+ \subseteq \Delta^0$ and $\Delta^- \subseteq \Delta^1$, the above is equivalent to (moving the brackets)
$(\Delta_Q \oplus \Delta_P) \setminus \Delta^- \cup \Delta^+; (\Gamma_Q \cup \Gamma_P) \setminus \Gamma^- \cup \Gamma^+ \vdash_{\mathrm{R}} P'$ which in turn is :
$\Delta \setminus \Delta^- \cup \Delta^+; \Gamma \setminus \Gamma^- \cup \Gamma^+ \vdash_{\mathrm{R}} P'$.
So (Γ and Δ)$^{(+\text{ and }-)}$ are preserved as well.

- (Res)
  Let $P \xrightarrow{\mu} P'$ and $\Theta \vdash_{\mathrm{R}} (\boldsymbol{\nu} a)\, P$ with $a \notin \mathrm{n}(\mu)$.
  Because restriction only hides a from Θ, Γ^0 and Δ^0 are preserved from P to $(\boldsymbol{\nu} a)\, P$, and so are Γ^+ and Δ^+.
  Concerning Δ^1 and Γ^1 there are two cases.
  First, if $\mu = \tau$ and, for $P \xrightarrow{\mu} P'$, $\Delta^1 = \Gamma^1 = \Delta^- = \Gamma^- \{a\}$ ($a \in \mathbf{N}_{\mathtt{l}}$), we have :
  $\Delta_{\mathtt{l}}, a; \Gamma_{\mathtt{l}}, a; \Delta_{\mathtt{u}}; \Gamma_{\mathtt{u}} \vdash_{\mathrm{R}} P, P'$, and $\Delta_{\mathtt{l}}; \Gamma_{\mathtt{l}}; \Delta_{\mathtt{u}}; \Gamma_{\mathtt{u}} \vdash_{\mathrm{R}} (\boldsymbol{\nu} a)\, P, (\boldsymbol{\nu} a)\, P'$.
  Basically, Before restricting, the transition was "consuming" the single input/output capability of a, and after restricting this change is hidden.
  So after restricting we have $\Gamma^1 = \Delta^1 = \Gamma^- = \Delta^- = \emptyset$ (the other sets still empty of course).
  Otherwise, we will have $a \notin (\Gamma^1 \cup \Delta^1)$ so it is preserved, and so are Γ^+ and Δ^1.
- (Inp)
  This rule produces a transition of the form :
  $P = a?\{\mathrm{l}_j(x_j){=}P_j \mid j{\in}J\} \xrightarrow{a?\mathrm{l}_k\langle b\rangle} P_k\{^b\!/\!_{x_k}\} = P'$,
  So we have to prove that $\Gamma^0_{\mathtt{l}} = \Gamma^+_{\mathtt{l}} = \{b\}_{\mathtt{l}}$, $\Gamma^0_{\mathtt{u}} = \Gamma^+_{\mathtt{u}} = \{b\}_{\mathtt{u}}$, $\Delta^1_{\mathtt{l}} = \Delta^-_{\mathtt{l}} = \{a\}_{\mathtt{l}}$, $\Delta^1_{\mathtt{u}} = \{a\}_{\mathtt{u}}$, knowing that $b \notin \Gamma_{\mathtt{l}}$.
  $\Gamma^0 = \{b\}_{\mathtt{r}}$ is precisely the side condition of 4.1.6.1.
  Rule (R-Inp) states that a typing of P will be as follows:
  $\{a\}_{\mathtt{l}}; \Gamma_{\mathtt{l}}; \emptyset; \Gamma_{\mathtt{u}} \vdash_{\mathrm{R}} P$ provided that (the substitution has been applied already in the following)
  $\emptyset; \Gamma_{\mathtt{l}}, \{b\}_{\mathtt{l}}; \emptyset; \Gamma_{\mathtt{u}}, \{b\}_{\mathtt{u}} \vdash_{\mathrm{R}} P'$.
  Because $a \notin \mathbf{N}_{\mathtt{u}}$ is a condition of (R-Inp), we have $\Delta^1_{\mathtt{u}} = \{a\}_{\mathtt{u}} = \emptyset \subseteq \Delta_{\mathtt{u}}$.
  The other required properties are clearly visible in these typings.

- (Rep)
  Let $P = a?^*\{\mathtt{l}_j(x_j){=}P_j \ \mid\ j{\in}J\} \xrightarrow{a?\mathtt{l}_k\langle b\rangle} P \mid P_k\{^b\!/_{x_k}\} = P'$.
  Like for (Inp), we have to prove $\Gamma^0_{\mathtt{l}} = \Gamma^+_{\mathtt{l}} = \{b\}_{\mathtt{l}}$, $\Gamma^0_{\mathtt{u}} = \Gamma^+_{\mathtt{u}} = \{b\}_{\mathtt{u}}$, $\Delta^1_{\mathtt{l}} = \Delta^-_{\mathtt{l}} = \{a\}_{\mathtt{l}}$, $\Delta^1_{\mathtt{u}} = \{a\}_{\mathtt{u}}$, knowing that $b \notin \Gamma_{\mathtt{l}}$.
  $\Gamma^0 = \{b\}_{\mathtt{r}}$ is immediate, as before.
  The typings of P and P' will be as follows (using (R-Rep) and (R-Par))
  $\emptyset; \Gamma_{\mathtt{l}}; \{a\}_{\mathtt{u}}; \Gamma_{\mathtt{u}} \vdash_{\mathrm{R}} P$
  $\emptyset; \Gamma_{\mathtt{l}}, \{b\}_{\mathtt{l}}; \{a\}_{\mathtt{u}}; \Gamma_{\mathtt{u}}, \{b\}_{\mathtt{u}} \vdash_{\mathrm{R}} P'$.
  Because $a \notin \mathbf{N}_{\mathtt{l}}$ is a condition of (R-Rep), we have $\Delta^1_{\mathtt{l}} = \{a\}_{\mathtt{l}} = \emptyset \subseteq \Delta_{\mathtt{l}}$.
  Again, the other required properties are clearly visible in these typings.

B.9 Replication Proposition (4.2.11)

- A first requirement is that all process pairs of the form given in the Proposition are complete and discreet.
  Using the conditions of the Proposition we can derive
  $\Delta_{P\mathtt{l}} \oplus \Delta_{Q\mathtt{l}}; \emptyset; \Delta_{P\mathtt{u}} \oplus \Delta_{Q\mathtt{u}}; \Gamma_{\mathtt{u}} \vdash_{\mathrm{R}} (\boldsymbol{\nu}u)\,(\,u \gg v \mid P \mid Q\,)$ and
  $\Delta_{P\mathtt{l}} \oplus \Delta_{Q\mathtt{l}}; \emptyset; \Delta_{P\mathtt{u}} \oplus \Delta_{Q\mathtt{u}}; \Gamma_{\mathtt{u}} \vdash_{\mathrm{R}} (\boldsymbol{\nu}u)\,(\,u \gg v \mid P\,)(\boldsymbol{\nu}u)\,(\,u \gg v \mid Q\,)$
  We also have $\Gamma_{\mathtt{u}} \subseteq \Delta_{P\mathtt{u}} \oplus \Delta_{Q\mathtt{u}}$
  Which implies that both processes are complete. They are also discreet because P and Q are, and the two above judgments did not make use of (R-Out).
- Directly applying Completeness and Discreetness Preservation Lemmas we see that for any complete and discreet processes P and Q, for any relation $\mathcal{R}$, the processes P' and Q' of clause 1 (or $(\boldsymbol{\nu}r)\,(\,r \gg t \mid P'\,)$ and $(\boldsymbol{\nu}r)\,(\,r \gg t \mid Q'\,)$ in clause 2) described by Proposition 4.2.8 will be complete and discreet as well.
- Now let $\mathcal{R}_0$ be the set of pairs of the form $\big((\boldsymbol{\nu}u)\,(\,u \gg v \mid P \mid Q\,)\ ;\ (\boldsymbol{\nu}u)\,(\,u \gg v \mid P\,) \mid (\boldsymbol{\nu}u)\,(\,u \gg v \mid Q\,)\big)$ where P and Q satisfy the conditions of the Proposition.
  Let $\mathcal{R} = \Big\{(X, Y) \mid \exists X_0, Y_0 | X \equiv X_0$ and $Y \equiv Y_0$ and $(X_0\ \mathcal{R}_0\ Y_0$ or $Y_0\ \mathcal{R}_0\ X_0)\Big\}$
  We will prove that $\mathcal{R}$ satisfies the conditions of Definition 4.2.8 where the initial $P\ \mathcal{R}\ Q$ is replaced by $P\ \mathcal{R}_0\ Q$, and then prove that this is enough for $\mathcal{R}$ to be a strong receptive bisimulation, which then directly

implies the proposition.

- We will look all possible transitions of X and Y where $X\ \mathcal{R}_0\ Y$ and P and Q are the corresponding sub-processes as defined above. By the symmetry of X and Y, for all below transitions P and Q can be exchanged.
  - If $P \xrightarrow{\mu} P'$ where $\mathrm{bn}(\mu) \cap (\mathrm{fn}(Q) \cup \{u, v\}) = \emptyset$ and $u \notin \mathrm{n}(\mu)$ (which are required by (Par) and (Res) for both X and Y), and μ matches clause 1 of 4.2.8,
    $(\boldsymbol{\nu} u)\,(\,u \gg v \mid P \mid Q\,) \xrightarrow{\mu} (\boldsymbol{\nu} u)\,(\,u \gg v \mid P' \mid Q\,)$ and
    $(\boldsymbol{\nu} u)\,(\,u \gg v \mid P\,) \mid (\boldsymbol{\nu} u)\,(\,u \gg v \mid Q\,) \xrightarrow{\mu}$
    $(\boldsymbol{\nu} u)\,(\,u \gg v \mid P'\,) \mid (\boldsymbol{\nu} u)\,(\,u \gg v \mid Q\,)$
  - If $P \xrightarrow{a?1\langle r\rangle} P'$ with r fresh and $a \neq u$,
    $(\boldsymbol{\nu} u)\,(\,u \gg v \mid P \mid Q\,) \xrightarrow{a?1\langle r\rangle} (\boldsymbol{\nu} u)\,(\,u \gg v \mid P' \mid Q\,)$.
    $(\boldsymbol{\nu} r)\,(r \gg t \mid ((\boldsymbol{\nu} u)\,(\,u \gg v \mid P' \mid Q\,))) \equiv$
    $(\boldsymbol{\nu} u)\,(\boldsymbol{\nu} r)\,(r \gg t \mid u \gg v \mid P' \mid Q) \equiv$
    $(\boldsymbol{\nu} u)\,(u \gg v \mid ((\boldsymbol{\nu} r)\,r \gg t \mid P') \mid Q)$
    We can do the same on the other member of the pair :
    $(\boldsymbol{\nu} u)\,(\,u \gg v \mid P\,) \mid (\boldsymbol{\nu} u)\,(\,u \gg v \mid Q\,) \xrightarrow{a?1\langle r\rangle}$
    $(\boldsymbol{\nu} u)\,(\,u \gg v \mid P'\,) \mid (\boldsymbol{\nu} u)\,(\,u \gg v \mid Q\,)$ (this whole process is the Q' of 4.2.8).
    Then,
    $(\boldsymbol{\nu} r)\,(r \gg t \mid ((\boldsymbol{\nu} u)\,(\,u \gg v \mid P'\,) \mid (\boldsymbol{\nu} u)\,(\,u \gg v \mid Q\,))) \equiv$
    $((\boldsymbol{\nu} u)\,(\boldsymbol{\nu} r)\,r \gg t \mid u \gg v \mid P'\,) \mid (\boldsymbol{\nu} u)\,(\,u \gg v \mid Q\,) \equiv$
    $((\boldsymbol{\nu} u)\,u \gg v \mid ((\boldsymbol{\nu} r)\,r \gg t \mid P'\,)) \mid (\boldsymbol{\nu} u)\,(\,u \gg v \mid Q\,)$.
    We see that this process and the one above are related by $\mathcal{R}_0$ so the ones before starting the sequences of $\equiv$ are related by $\mathcal{R}$, which is what was requested for this case.
  - If $P \xrightarrow{(\boldsymbol{\nu} x)\,c!1\langle x\rangle} P'$ and $Q \xrightarrow{c?1\langle x\rangle} Q'$, and $x \notin \mathrm{fn}(Q)$, then
    $(\boldsymbol{\nu} u)\,(\,u \gg v \mid P \mid Q\,) \xrightarrow{\tau} (\boldsymbol{\nu} u)\,(\,u \gg v \mid P' \mid Q'\,)$ and $(\boldsymbol{\nu} u)\,(\,u \gg v \mid P\,) \mid (\boldsymbol{\nu} u)\,(\,u \gg v|Q\,) \xrightarrow{\tau} (\boldsymbol{\nu} u)\,(\,u \gg v|P'\,) \mid (\boldsymbol{\nu} u)\,(\,u \gg v \mid Q'\,)$
  - If $P \xrightarrow{(\boldsymbol{\nu} x)\,u!1\langle x\rangle} P'$ and $x \notin \{u, v\}$ then
    $(\boldsymbol{\nu} u)\,(\,u \gg v \mid P \mid Q\,) \xrightarrow{\tau} (\boldsymbol{\nu} u)\,(((\boldsymbol{\nu} x)\,u \gg v|v!1(\boldsymbol{\nu} z).z \gg x|P')|Q\,)$
    which is structurally congruent to (having $x \notin \mathrm{fn}(u \gg v)$)

$(\boldsymbol{\nu} u)\,((u \gg v \mid (\boldsymbol{\nu} x)\,(v!\mathrm{l}(\boldsymbol{\nu} z).z \gg x \mid P'))|Q\,)$
For the other member of the pair,
$(\boldsymbol{\nu} u)\,(\,u \gg v \mid P\,) \mid (\boldsymbol{\nu} u)\,(\,u \gg v \mid Q\,) \xrightarrow{\tau}$
$(\boldsymbol{\nu} u)\,((\boldsymbol{\nu} x)\,u \gg v \mid v!\mathrm{l}(\boldsymbol{\nu} z).z \gg x \mid P'\,) \mid (\boldsymbol{\nu} u)\,(\,u \gg v \mid Q\,) \equiv$
$((\boldsymbol{\nu} u)\,u \gg v \mid (\boldsymbol{\nu} x)\,v!\mathrm{l}(\boldsymbol{\nu} z).z \gg x \mid P') \mid ((\boldsymbol{\nu} u)\,(\,u \gg v \mid Q\,))$
Therefore, in both cases, P was transformed into $(\boldsymbol{\nu} x)\,v!\mathrm{l}(\boldsymbol{\nu} z).z \gg x|P'$.

- We still have to prove that the weakened version of Definition 4.2.8 used is enough for having $X \sim_{\mathrm{R}} Y$. For this, knowing that $\mathcal{R}_0$ satisfies that weakened condition, $\mathcal{R}$ indeed is a receptive bisimulation. This follows quite easily from $X \equiv Y$ and $X \xrightarrow{\mu} X'$ implying that $\exists Y'(Y \xrightarrow{\mu} Y'$ and $X' \equiv Y')$ and $\forall C[\cdot](C[X] \equiv C[Y])$:
  Let $X\,\mathcal{R}\,Y$. Then there are X_0 and Y_0 such that $X \equiv X_0\,\mathcal{R}_0\,Y_0 \equiv Y$.
  - If $X \xrightarrow{\mu} X'$ with μ as specified in clause 1 of 4.2.8, then, from $X \equiv X_0$ there is X_0' such that $X_0 \xrightarrow{\mu} X_0'$ and $X' \equiv X_0'$.
    From $X_0\,\mathcal{R}_0\,Y_0$ and $X_0 \xrightarrow{\mu} X_0'$, there is Y_0' such that $Y_0 \xrightarrow{\mu} Y_0'$ and $X_0'\,\mathcal{R}\,Y_0'$ (note that it is $\mathcal{R}$, not $\mathcal{R}_0$).
    From $Y_0 \equiv Y$ and $Y_0 \xrightarrow{\mu} Y_0'$, there is Y' such that $Y \xrightarrow{\mu} Y_0'$ and $Y_0' \equiv Y'$.
    $X' \equiv X_0'\,\mathcal{R}\,Y_0' \equiv Y'$ implies $X'\,\mathcal{R}\,Y'$.
  - If $X \xrightarrow{\mu} X'$ with μ as specified in clause 2, then, from $X \equiv X_0$ and $X \xrightarrow{\mu} X'$, there is X_0' such that $X_0 \xrightarrow{\mu} X_0'$ and $(\boldsymbol{\nu} r)\,(r \gg t \mid X') \equiv (\boldsymbol{\nu} r)\,(r \gg t \mid X_0')$.
    From $X_0\,\mathcal{R}_0\,Y_0$ and $X_0 \xrightarrow{\mu} X_0'$, there is Y_0' such that $Y_0 \xrightarrow{\mu} Y_0'$ and $(\boldsymbol{\nu} r)\,(r \gg t \mid X_0')\,\mathcal{R}\,(\boldsymbol{\nu} r)\,(r \gg t \mid Y_0')$.
    Then, there is Y' such that $Y \xrightarrow{\mu} Y'$ and $Y_0' \equiv Y'$.
    So $(\boldsymbol{\nu} r)\,(r \gg t \mid X')\,\mathcal{R}\,(\boldsymbol{\nu} r)\,(r \gg t \mid Y')$.

B.10 π_{a}^{V}- TyCO Encoding Typability (Prop. 5.2.2.1)

Lemma B.10.1. *If* $\Sigma \vdash_{\mathrm{S}} v{:}T$ *then* $[\![\,\Sigma\,]\!], u{:}[\![\,T\,]\!] \vdash_{\mathrm{S}} [\![\,v\,]\!]_u$.

Proof. We will work by induction on v.

Let $\Sigma \vdash_{\mathrm{S}} v = a{:}T$ be a simple name.

Then $[\![\,v\,]\!]_u = u \gg v$ and $[\![\,\Sigma\,]\!], u{:}[\![\,T\,]\!] \vdash_{\mathrm{S}} v{:}[\![\,T\,]\!], u{:}[\![\,T\,]\!]$ so $[\![\,\Sigma\,]\!], u{:}[\![\,T\,]\!] \vdash_{\mathrm{S}} [\![\,v\,]\!]_u$.

Now let $\Sigma \vdash_{\mathrm{S}} v = \mathrm{l_k}\langle w\rangle{:}\{\mathrm{l}_j{:}T_j \mid j{\in}J\}$ (where $k \in J$ and $\Sigma \vdash_{\mathrm{S}} w{:}T_k$).
Then $[\![v]\!]_u = u?^*\{\mathrm{d}(r){=}r!\mathrm{l_k}(\boldsymbol{\nu} u').[\![w]\!]_{u'}\}$ and
$[\![\Sigma]\!] \vdash_{\mathrm{S}} v{:}[\{\mathrm{d}{:}[\{\mathrm{l}_j{:}[\![T_j]\!] \mid j{\in}J\}]\}]$.
We have $[\![\Sigma]\!], x{:}[\{\{\mathrm{l}_j{:}[\![T_j]\!] \mid j{\in}J\}\}], u{:}[\![T]\!], \vdash_{\mathrm{S}} w{:}[\![T_k]\!]$ (the additional hypothesis will be used below; Adding an entry in the hypothesis of a derivable typing judgment will keep it correct) so by induction hypothesis,
$[\![\Sigma]\!], x{:}[\{\mathrm{l}_j{:}[\![T_j]\!] \mid j{\in}J\}], u{:}[\![T]\!], u'{:}[\![T_k]\!] \vdash_{\mathrm{S}} [\![w]\!]_{u'}$.
The (ST-Bout) rule yields
$[\![\Sigma]\!], x{:}[\{\mathrm{l}_j{:}[\![T_j]\!] \mid j{\in}J\}], u{:}[\![T]\!] \vdash_{\mathrm{S}} x!\mathrm{l_k}(\boldsymbol{\nu} u').[\![w]\!]_{u'}$
Then applying (ST-Rep) with a set J containing a single index j for which $\mathrm{l}_j = \mathrm{d}$:
$[\![\Sigma]\!], u{:}[\![T]\!] \vdash_{\mathrm{S}} u?^*\{\mathrm{d}(r){=}r!\mathrm{l_k}(\boldsymbol{\nu} u').[\![w]\!]_{u'}\}$, which is $[\![v]\!]_u$.

$\square$

The proof of 5.2.2.1 itself will work by induction on the process structure.

- $P = \mathbf{0}$ is trivial.
- $P = P_1 \mid P_2$. $\Sigma \vdash_{\mathrm{S}} P$ has to be derived from $\Sigma \vdash_{\mathrm{S}} P_i$, for both $i = 1, 2$. By induction hypothesis, $[\![\Sigma]\!] \vdash_{\mathrm{S}} [\![P_i]\!]$ for both $i = 1, 2$ so $\Sigma \vdash_{\mathrm{S}} [\![P_1]\!] \mid [\![P_2]\!] = [\![P_1 \mid P_2]\!]$.
- $P = (\boldsymbol{\nu} a)\, P_1$. $\Sigma \vdash_{\mathrm{S}} (\boldsymbol{\nu} a)\, P_1$ implies (through (S-Res)) there is T such that $\Sigma, a{:}T \vdash_{\mathrm{S}} P_1$ so by induction $[\![\Sigma]\!], a{:}[\![T]\!] \vdash_{\mathrm{S}} [\![P_1]\!]$ and applying (S-Res) again $[\![\Sigma]\!] \vdash_{\mathrm{S}} (\boldsymbol{\nu} a)\, [\![P]\!] = [\![(\boldsymbol{\nu} a)\, P]\!]$.
- $P = a!v$. $\Sigma \vdash_{\mathrm{S}} P$ implies there is T such that $\Sigma \vdash_{\mathrm{S}} (a{:}[T]$ and $v{:}T)$.
The type encoding rules tell that $[\![\Sigma]\!] \vdash_{\mathrm{S}} a{:}[\{\mathrm{c}{:}[\![T]\!]\}], v{:}[\![T]\!]$.
Lemma B.10.1 implies that $[\![\Sigma]\!], u{:}[\![T]\!] \vdash_{\mathrm{S}} [\![v]\!]_u$.
So by (S-Bout) $[\![\Sigma]\!] \vdash_{\mathrm{S}} a!\mathrm{c}(\boldsymbol{\nu} u).[\![v]\!]_u$ which is $[\![a!v]\!]$.
- $P = a?(x).P_1$.
So we have a T such that $\Sigma \vdash_{\mathrm{S}} a{:}[T]$ and $\Sigma, x{:}T \vdash_{\mathrm{S}} P_1$.
$[\![\Sigma]\!] \vdash_{\mathrm{S}} a{:}[\{\mathrm{t_a}{:}[\![T]\!]\}]$.
By induction, $[\![\Sigma]\!], x{:}[\![T]\!] \vdash_{\mathrm{S}} [\![P_1]\!]$.
So $[\![\Sigma]\!] \vdash_{\mathrm{S}} a?\{\mathrm{c}(x){=}[\![P_1]\!]\}$ which is $[\![P]\!]$.
- $P = a?^*(x).P_1$: The proof is identical to the above one, only using (ST-Rep) instead of (ST-Vout).
- $P = \mathsf{case}\, v\, \mathsf{of}\, \{\mathrm{l}_j(x_j){=}P_j \mid j{\in}J\}$.
Then we have $T = \{\mathrm{l}_j{:}T_j \mid j{\in}J\}$ such that $\Sigma \vdash_{\mathrm{S}} v{:}T$. We have to prove the typability of $(\boldsymbol{\nu} u)\left([\![v]\!]_u \mid u!\mathrm{d}(\boldsymbol{\nu} r).r?\{\mathrm{l}_j(x_j){=}[\![P_j]\!] \mid j{\in}J\} \right)$ under $[\![\Sigma]\!]$.

By (SP-Case), $\Sigma, x{:}T_j \vdash_{\mathrm{S}} P_j$, for all $j \in J$.
By induction (and adding types for bound names),
$[\![\Sigma]\!], u{:}[\![T]\!], r{:}[\{\mathrm{l}_j{:}[\![T_j]\!] \mid j{\in}J\}], x_j{:}[\![T_j]\!] \vdash_{\mathrm{S}} [\![P_j]\!]$ for all $j \in J$.
By (ST-Vinp) :
$[\![\Sigma]\!], u{:}[\![T]\!], r{:}[\{\mathrm{l}_j{:}[\![T_j]\!] \mid j{\in}J\}] \vdash_{\mathrm{S}} r?\{\mathrm{l}_j(x_j){=}[\![P_j]\!] \mid j{\in}J\}$.
$[\![T]\!]$ being $[\{\mathrm{d}{:}[\{\mathrm{l}_j{:}[\![T_j]\!] \mid j{\in}J\}]\}]$, we can apply (ST-Bout) :
$[\![\Sigma]\!], u{:}[\![T]\!] \vdash_{\mathrm{S}} u!\mathrm{d}(\boldsymbol{\nu}r).r?\{\mathrm{l}_j(x_j){=}[\![P_j]\!] \mid j{\in}J\}$.
Because $\Sigma \vdash_{\mathrm{S}} v{:}T$, $[\![\Sigma]\!] \vdash_{\mathrm{S}} v{:}[\![T]\!]$. Lemma B.10.1 then gives
$[\![\Sigma]\!], u{:}[\![T]\!] \vdash_{\mathrm{S}} [\![v]\!]_u$.
So we can apply (S-Par) :
$[\![\Sigma]\!], u{:}[\![T]\!] \vdash_{\mathrm{S}} [\![v]\!]_u \mid u!\mathrm{d}(\boldsymbol{\nu}r).r?\{\mathrm{l}_j(x_j){=}[\![P_j]\!] \mid j{\in}J\}$.
Finally, (S-Res) :
$[\![\Sigma]\!] \vdash_{\mathrm{S}} [\![P]\!]$.

B.11 Substitution part I (Lemma 5.2.5.1)

Let $\mathcal{R}$ be the relation expressed by the lemma ($X \mathrel{\mathcal{R}} Y$ iff there are $a \in \mathbf{N}_{\mathrm{u}}$, $u \in \mathbf{N}_{\mathrm{u}}$ and $v \in \mathbf{N}_{\mathrm{p}}$ such that $X = [\![v]\!]_a$ and $Y = (\boldsymbol{\nu}u)\,(a \gg u \mid [\![v]\!]_u)$).

Having $P \mathrel{\mathcal{R}} Q$ and $P \xrightarrow{\mu} P'$, and if μ is an input with receptive object r then $P'' = (\boldsymbol{\nu}r)\,(r \gg t \mid P')$ otherwise $P'' = P'$, we prove :

There are μ_2 and $C[\cdot]$ s.t. $P'' \xrightarrow{\mu_2} (P \mid C[P'''])$, $Q \xrightarrow{\mu} Q'$, $Q'' \overset{\mu_2}{\Longrightarrow} (Q \mid C[Q'''])$ with $P''' \mathrel{\mathcal{R}} Q'''$. (Q'' being Q' with a trigger name if $\mu = a?\mathrm{l}\langle r\rangle$)

Similarly if $Q \xrightarrow{\mu} Q'$ then $P \xrightarrow{\mu} P'$, $P'' \xrightarrow{\mu_2} (P \mid C[P'''])$ and $Q'' \overset{\mu_2}{\Longrightarrow} (Q \mid C[Q'''])$ with $P''' \mathrel{\mathcal{R}} Q'''$, P'' and Q'' being defined as above. (Note that in both cases the weak transitions are on the Q side only).

The additional μ_2 and τ transition sequences are independent, which means that several instances of these transition sequences can be running simultaneously without interfering with each other, and they are deterministic, which means that once μ occurred, that sequence of transition is the only one available, i.e. no "choice" is made in it.

We will afterwards prove that for any relation $\mathcal{R}$ satisfying these properties $P \mathrel{\mathcal{R}} Q$ implies $Q \gtrsim_{\mathrm{R}} P$.

The lemma has two cases, whether v is a simple name a or a labeled value $\mathrm{l}\langle w\rangle$.

We will first prove the case where v is a simple name.

Let $P = a \gg v$ and $Q = (\boldsymbol{\nu}u)\,(a \gg u \mid u \gg v)$.

The only transition provided by either process is an input at a :

First the case of a plain name received on a.

By definition of dynamic links :

$P \xrightarrow{a?\mathrm{d}\langle c\rangle} P \mid v!\mathrm{d}(\boldsymbol{\nu}y).y \gg c \xrightarrow{v!\mathrm{d}(\boldsymbol{\nu}y)} P \mid y \gg c$

On the Q side :

$Q \xrightarrow{a?\mathrm{d}\langle c\rangle} (\boldsymbol{\nu}u)\,(a \gg u \mid u \gg v \mid u!\mathrm{d}(\boldsymbol{\nu}t).t \gg c) \xrightarrow{\tau}\equiv$
$(\boldsymbol{\nu}u)\,(a \gg u \mid u \gg v) \mid (\boldsymbol{\nu}t)\,(v!\mathrm{d}(\boldsymbol{\nu}y).y \gg t \mid t \gg c) =$
$Q \mid v!\mathrm{d}(\boldsymbol{\nu}y).(\boldsymbol{\nu}t)\,(y \gg t \mid t \gg c) \xrightarrow{v!\mathrm{d}(\boldsymbol{\nu}y)} Q \mid (\boldsymbol{\nu}t)\,(y \gg t \mid t \gg c)$.

And we have $y \gg c \;\mathcal{R}\; (\boldsymbol{\nu}t)\,(y \gg t \mid t \gg c)$ as required (using a simple context $C[\cdot] = [\cdot]$)

Now the case where a receptive name is sent on a :

$P \xrightarrow{a?\mathrm{d}\langle x\rangle} P \mid v!\mathrm{d}(\boldsymbol{\nu}y).y \gg x.$

We need to put a trigger name (t fresh and plain) to make that process complete, as specified by 4.2.7.3 :

$(\boldsymbol{\nu}x)\,P \mid v!\mathrm{d}(\boldsymbol{\nu}y).y \gg x \mid x \gg t \xrightarrow{v!\mathrm{d}(\boldsymbol{\nu}y)}\equiv P \mid (\boldsymbol{\nu}x)\,y \gg x \mid x \gg t = P \mid C[y \gg x]$, where $C[\cdot] = (\boldsymbol{\nu}x)\,([\cdot] \mid x \gg t)$

Concerning Q :

$Q \xrightarrow{a?\mathrm{d}\langle x\rangle} (\boldsymbol{\nu}u)\,(a \gg u \mid u \gg v \mid u!\mathrm{d}(\boldsymbol{\nu}z).z \gg x)$. Adding link to t :
$(\boldsymbol{\nu}x)\,(\boldsymbol{\nu}u)\,(a \gg u \mid u \gg v \mid u!\mathrm{d}(\boldsymbol{\nu}z).z \gg x \mid x \gg t) \xrightarrow{\tau}$
$(\boldsymbol{\nu}x)\,(\boldsymbol{\nu}u)\,(a \gg u \mid u \gg v \mid v!\mathrm{d}(\boldsymbol{\nu}y).(\boldsymbol{\nu}z)\,y \gg z \mid z \gg x \mid x \gg t) \xrightarrow{v!\mathrm{d}(\boldsymbol{\nu}y)}$
$(\boldsymbol{\nu}u)\,(a \gg u \mid u \gg v) \mid (\boldsymbol{\nu}z)\,(\boldsymbol{\nu}x)\,(y \gg z \mid z \gg x \mid x \gg t)$
which can be rewritten ($\equiv$) as
$Q \mid (\boldsymbol{\nu}x)\,((\boldsymbol{\nu}z)\,y \gg z \mid z \gg x) \mid x \gg t =$
$Q \mid C[((\boldsymbol{\nu}z)\,y \gg z \mid z \gg x)]$

We have $y \gg x \;\mathcal{R}\; (\boldsymbol{\nu}z)\,(y \gg z \mid z \gg x)$ which is what we required.

Now the case where v is a labeled value. Let $v = \mathrm{l}\langle w\rangle$.

We have $P = a?^*\{\mathrm{d}(x){=}x!\mathrm{l}(\boldsymbol{\nu}y).[\![\, w \,]\!]_y\}$ and
$Q = (\boldsymbol{\nu}u)\,(a \gg u \mid u?^*\{\mathrm{d}(x){=}x!\mathrm{l}(\boldsymbol{\nu}z).[\![\, w \,]\!]_z\})$.

The only available transition on either process is an input at a. First the case where a plain name is transmitted.

on the P-side :

$P \xrightarrow{a?\mathrm{d}\langle c\rangle} P \mid c!\mathrm{l}(\boldsymbol{\nu}y).[\![\, w \,]\!]_y \xrightarrow{c!\mathrm{l}(\boldsymbol{\nu}y)} P \mid [\![\, w \,]\!]_y$

Concerning Q :

$P \xrightarrow{a?\mathrm{d}\langle c\rangle} (\boldsymbol{\nu}u)\,(a \gg u \mid u?^*\{\mathrm{d}(x){=}x!\mathrm{l}(\boldsymbol{\nu}z).[\![\, w \,]\!]_z\} \mid u!\mathrm{d}(\boldsymbol{\nu}r).r \gg c) \xrightarrow{\tau}$
(Note that $r \in \mathbf{N}_1$)

Transmitting r on u (and renaming z to z' to avoid duplicates) :

$(\boldsymbol{\nu} ru)\,(a \gg u \mid u?^*\{\mathrm{d}(x){=}x!\mathrm{l}(\boldsymbol{\nu} z).[\![\, w \,]\!]_z\} \mid r!\mathrm{l}(\boldsymbol{\nu} z').[\![\, w \,]\!]_{z'} \mid r \gg c) \xrightarrow{\tau}$
Transmitting z' on r :
$(\boldsymbol{\nu} z'ru)\,(a \gg u \mid u?^*\{\mathrm{d}(x){=}x!\mathrm{l}(\boldsymbol{\nu} z).[\![\, w \,]\!]_z\} \mid c!\mathrm{l}(\boldsymbol{\nu} y).y \gg z' \mid [\![\, w \,]\!]_{z'})$
$\xrightarrow{c!\mathrm{l}(\boldsymbol{\nu} y)}$
$(\boldsymbol{\nu} z'ru)\,(a \gg u \mid u?^*\{\mathrm{d}(x){=}x!\mathrm{l}(\boldsymbol{\nu} z).[\![\, w \,]\!]_z\} \mid y \gg z' \mid [\![\, w \,]\!]_{z'})$.

r has been “consumed” so its restriction can be removed through $\equiv$. Also moving the brackets to reach an interesting result we get :

$Q \mid ((\boldsymbol{\nu} z')\, y \gg z' \mid [\![\, w \,]\!]_{z'})$.

The right component of the composition is related by $\mathcal{R}$ with the corresponding one for P.

The development in case of a receptive name received on a is very similar, there will just be one additional forwarding, much like in the case $v = a$.

Here is now an outline of a proof that the proof technique we used was valid.

Let $\widetilde{\mathcal{R}}$ be the set of pairs

$(C_1[P_1] \mid C_2[P_2] \mid \ \ldots \ \mid C_n[P_n] \ ; \ C_1[Q_1] \mid C_2[Q_2] \mid \ \ldots \ \mid C_n[Q_n])$

where $\forall 1 \leq i \leq n$, $P_i \;\mathcal{R}\; Q_i$. We require the contexts to be of the form $(\boldsymbol{\nu}\tilde{x})\,([\cdot] \mid \tilde{x} \gg \tilde{t})$.

Let $X \widetilde{\mathcal{R}} Y$ and $X \xrightarrow{\mu} X'$

We won't handle the case of an interaction between two terms of the parallel composition because each term listens on a name that was created fresh.

Now, if this transition happens inside $C_i[P_i]$ then we can apply the properties of $\mathcal{R}$ and see that we fall back on an $(X'; Y')$ pair of $\widetilde{\mathcal{R}}$ because combining two contexts of the above form yields one context of the same form.

B.12 Substitution part II (Lemma 5.2.5.2)

This is proven by induction on the structure of P.

We will assume that $x \in \mathrm{fn}(P)$ otherwise neither substitution does anything and we'd just have to prove $(\boldsymbol{\nu} u)\,([\![\, P \,]\!] \mid [\![\, b \,]\!]_u) \gtrsim_{\mathrm{R}} [\![\, P \,]\!]$. Because $u \notin \mathrm{fn}([\![\, P \,]\!])$ (as u is uniform) and u is restricted, $[\![\, b \,]\!]_u$ couldn't be triggered so we'd have $\sim_{\mathrm{R}}$.

- $P = \mathbf{0}$
  Because neither $(\boldsymbol{\nu} u)\,(\mathbf{0} \mid [\![\, b \,]\!]_u)$ nor $\mathbf{0}$ have any transition, the first one trivially receptive-expands the second one.

- $P = P_1|P_2$.
  Using the replication proposition (5.2.4) we get :
  $(\boldsymbol{\nu} u)\,(\llbracket P_1|P_2 \rrbracket\{^u\!/\!_x\} \mid \llbracket b \rrbracket_u) =$
  $(\boldsymbol{\nu} u)\,((\llbracket P_1 \rrbracket\{^u\!/\!_x\}|\llbracket P_2 \rrbracket\{^u\!/\!_x\}) \mid \llbracket b \rrbracket_u) \sim_{\mathrm{R}}$
  $((\boldsymbol{\nu} u)\,(\llbracket P_1 \rrbracket\{^u\!/\!_x\} \mid \llbracket b \rrbracket_u)) \mid ((\boldsymbol{\nu} u)\,(\llbracket P_2 \rrbracket\{^u\!/\!_x\} \mid \llbracket b \rrbracket_u))$.
  We assume (by induction) that the lemma is true for both P_i :
  $(\boldsymbol{\nu} u)\,(\llbracket P_i \rrbracket\{^u\!/\!_x\} \mid \llbracket b \rrbracket_u) \gtrsim_{\mathrm{R}} \llbracket P_i\{^b\!/\!_x\} \rrbracket$.
  So, applying 4.2.10 twice on the expression above :
  $((\boldsymbol{\nu} u)\,(\llbracket P_1 \rrbracket\{^u\!/\!_x\} \mid \llbracket b \rrbracket_u)) \mid ((\boldsymbol{\nu} u)\,(\llbracket P_2 \rrbracket\{^u\!/\!_x\} \mid \llbracket b \rrbracket_u)) \gtrsim_{\mathrm{R}}$
  $(\llbracket P_1\{^b\!/\!_x\} \rrbracket) \mid ((\boldsymbol{\nu} u)\,(\llbracket P_2 \rrbracket\{^u\!/\!_x\} \mid \llbracket b \rrbracket_u)) \gtrsim_{\mathrm{R}}$
  $(\llbracket P_1\{^b\!/\!_x\} \rrbracket) \mid (\llbracket P_2\{^b\!/\!_x\} \rrbracket) =$
  $\llbracket (P_1 \mid P_2)\{^b\!/\!_x\} \rrbracket = \llbracket P\{^b\!/\!_x\} \rrbracket$.
  Because we have $X \sim_{\mathrm{R}}\gtrsim_{\mathrm{R}}\gtrsim_{\mathrm{R}} Y$ implies $X \gtrsim_{\mathrm{R}} Y$ the lemma is proven for this case.
- $P = (\boldsymbol{\nu} a)\,P_0$
  $x \in \mathrm{fn}(P)$ implies $x \neq a$.
  If $b = a$ then alpha-renaming can be used (by definition of substitution) to avoid capture and we fall back on the general case. So, assuming $b \neq a$:
  By induction we suppose that the lemma is true for P_0.
  $(\boldsymbol{\nu} u)\,(\llbracket P_0 \rrbracket\{^u\!/\!_x\} \mid \llbracket b \rrbracket_u) \gtrsim_{\mathrm{R}} \llbracket (P_0\{^b\!/\!_x\}) \rrbracket$
  The congruence properties give :
  $(\boldsymbol{\nu} a)\,(\boldsymbol{\nu} u)\,(\llbracket P_0 \rrbracket\{^u\!/\!_x\} \mid \llbracket b \rrbracket_u) \gtrsim_{\mathrm{R}} (\boldsymbol{\nu} a)\,\llbracket P_0\{^b\!/\!_x\} \rrbracket$
  We have $a \in \mathbf{N}_{\mathtt{p}}$ and $u \in \mathbf{N}_{\mathtt{u}}$ so $a \neq u$ and we also have $a \notin \{b, x\}$ so the restriction can be pushed inside :
  $(\boldsymbol{\nu} u)\,(\llbracket (\boldsymbol{\nu} a)\,P_0 \rrbracket\{^u\!/\!_x\} \mid \llbracket b \rrbracket_u) \gtrsim_{\mathrm{R}} (\boldsymbol{\nu} a)\,\llbracket ((\boldsymbol{\nu} a)\,P_0)\{^b\!/\!_x\} \rrbracket$.
- $P = a!v$
  $x \in \mathrm{fn}(P)$ implies either $x = a$ or $x \in \mathrm{n}(v)$.
  If $x = a$ (In which case b has to be a simple name) :
  we want to prove
  $(\boldsymbol{\nu} u)\,(\llbracket a!v \rrbracket\{^u\!/\!_a\}|u \gg b) \gtrsim_{\mathrm{R}} \llbracket b!v \rrbracket$
  Applying the encoding (calling X and Y the two sides of the expression) :
  $X = (\boldsymbol{\nu} u)\,(u!\mathrm{c}(\boldsymbol{\nu} s).\llbracket v \rrbracket_s \mid u \gg b) \gtrsim_{\mathrm{R}} b!\mathrm{c}(\boldsymbol{\nu} r).\llbracket v \rrbracket_r = Y$.
  X has exactly one transition which is to send s over u which yields $(\boldsymbol{\nu} u)\,(b!\mathrm{c}(\boldsymbol{\nu} r).\llbracket v \rrbracket_r \mid u \gg b)$ where the forwarding can't be activated anymore so we have

$X \xrightarrow{\tau}\sim_{\mathrm{R}} Y$ which implies $X \gtrsim_{\mathrm{R}} Y$ as required.
We now suppose $x \in \mathrm{n}(v)$ (which contains exactly one name, ie we have $v = \mathrm{l}_1\langle \cdots \mathrm{l}_\mathrm{n}\langle x\rangle\rangle$)
We want to prove $(\boldsymbol{\nu} u)\,(\llbracket\, a!v \,\rrbracket\{^u/_x\} \mid \llbracket\, b \,\rrbracket_u) \gtrsim_{\mathrm{R}} \llbracket\, a!v\{^b/_x\} \,\rrbracket$.
Applying the encoding :
$(\boldsymbol{\nu} u)\,(a!\mathrm{c}(\boldsymbol{\nu} r).\llbracket\, v \,\rrbracket_r\{^u/_x\} \mid \llbracket\, b \,\rrbracket_u) \gtrsim_{\mathrm{R}} a!\mathrm{c}(\boldsymbol{\nu} r).\llbracket\, v\{^b/_x\} \,\rrbracket_r$.
Both sides have exactly one transition which is $\xrightarrow{a!\mathrm{c}(\boldsymbol{\nu} r)}$ and leads to :

$$(\boldsymbol{\nu} u)\,(\llbracket\, v \,\rrbracket_r\{^u/_x\} \mid \llbracket\, b \,\rrbracket_u) \gtrsim_{\mathrm{R}} \llbracket\, v\{^b/_x\} \,\rrbracket_r \tag{1}$$

Let r' be that fresh name that is used as a private reference to x in $\llbracket\, v \,\rrbracket_r$ (i.e. the last step of the development of $\llbracket\, v \,\rrbracket_r$ is $\llbracket\, x \,\rrbracket_{r'}$, and let $C_v[\cdot]$ be the context defined by $\llbracket\, v\{^{[\cdot]}/_x\} \,\rrbracket_r$ where $\llbracket\, [\cdot] \,\rrbracket_{r'}$ is $[\cdot]$.
We then have $\llbracket\, v \,\rrbracket_r = C_v[r' \gg x]$.
(1) can then be rewritten as

$$(\boldsymbol{\nu} u)\,(C_v[r' \gg u] \mid \llbracket\, b \,\rrbracket_u) \gtrsim_{\mathrm{R}} C_v[\llbracket\, b \,\rrbracket_{r'}] \tag{2}$$

The left hand side process is bisimilar to the corresponding one where $\llbracket\, b \,\rrbracket_u$ is moved inside the context, as we show now.
If $v = x$ (ie v is a simple name) then $C_v[\cdot] = [\cdot]$ and there is nothing to change.
If $v = \mathrm{l}\langle w\rangle$ for some w, we have
$(\boldsymbol{\nu} u)\,(C_v[r' \gg u] \mid \llbracket\, b \,\rrbracket_u) = (\boldsymbol{\nu} u)\,(r?^*\{\mathrm{d}(a){=}a!\mathrm{l}(\boldsymbol{\nu} z).\llbracket\, w \,\rrbracket_z\{^u/_x\}\} \mid \llbracket\, b \,\rrbracket_u)$.
Because there can only be input at u once there has been input at r, we can move the restriction and $\llbracket\, b \,\rrbracket_u$ after the input prefix :
$(\boldsymbol{\nu} u)\,(r?^*\{\mathrm{d}(a){=}a!\mathrm{l}(\boldsymbol{\nu} z).\llbracket\, w \,\rrbracket_z\{^u/_x\}\} \mid \llbracket\, b \,\rrbracket_u) \sim_{\mathrm{R}}$
$r?^*\{\mathrm{d}(a){=}a!\mathrm{l}(\boldsymbol{\nu} z).(\boldsymbol{\nu} u)\,(\llbracket\, w \,\rrbracket_z\{^u/_x\} \mid \llbracket\, b \,\rrbracket_u)\}$.
Doing the same operation for each layer (each corresponding to one label in v) we have
$(\boldsymbol{\nu} u)\,(C_v[r' \gg u] \mid \llbracket\, b \,\rrbracket_u) \sim_{\mathrm{R}} C_v[(\boldsymbol{\nu} u)\, r' \gg u \mid \llbracket\, b \,\rrbracket_u]$.
Applying it on (2) :
$C_v[(\boldsymbol{\nu} u)\,(r' \gg u \mid \llbracket\, b \,\rrbracket_u)] \gtrsim_{\mathrm{R}} C_v[\llbracket\, b \,\rrbracket_{r'}]$
By Lemma 4.2.10, this is true if the parameters of $C_v[\cdot]$ are related by $\gtrsim_{\mathrm{R}}$:
$(\boldsymbol{\nu} u)\,(r' \gg u \mid \llbracket\, b \,\rrbracket_u) \gtrsim_{\mathrm{R}} \llbracket\, b \,\rrbracket_{r'}$.
This is true by 5.2.5.1

- $P = a?(v).P_0$

We have $a \notin \{x, v\}$ because by hypothesis $a \in \mathrm{fn}(P)$ and P does not receive on x.
We want to prove the following :
$(\boldsymbol{\nu} u)\,(a?\{\mathrm{c}\,(v){=}[\![\, P_0 \,]\!]\{{}^{u}\!/\!{}_{x}\}\} \mid [\![\, b \,]\!]_u) \gtrsim_{\mathrm{R}} a?\{\mathrm{c}\,(v){=}[\![\, P_0\{{}^{b}\!/\!{}_{x}\} \,]\!]\}$
Similarly to the other cases we can move $[\![\, b \,]\!]_u$ inside the input expression because no output at u can be done before any input at a :
$a?\{\mathrm{c}\,(v){=}(\boldsymbol{\nu} u)\,([\![\, P_0 \,]\!]\{{}^{u}\!/\!{}_{x}\} \mid [\![\, b \,]\!]_u)\} \gtrsim_{\mathrm{R}} a?\{\mathrm{c}\,(v){=}[\![\, P_0\{{}^{b}\!/\!{}_{x}\} \,]\!]\}$.
By 4.2.10 this is true if the same without the input prefix is true :
$(\boldsymbol{\nu} u)\,([\![\, P_0 \,]\!]\{{}^{u}\!/\!{}_{x}\} \mid [\![\, b \,]\!]_u) \gtrsim_{\mathrm{R}} [\![\, P_0\{{}^{b}\!/\!{}_{x}\} \,]\!]$. This is our induction hypothesis.

- $P = \mathsf{case}\, v\, \mathsf{of}\, \{\mathrm{l}_j(x_j){=}P_j \;\mid\; j{\in}J\}$
  We want to prove :
  $(\boldsymbol{\nu} u)\,\big((\boldsymbol{\nu} t)\,[\![\, v \,]\!]_t \mid t!\mathrm{d}\,(\boldsymbol{\nu} a).a?\{\mathrm{l}_j(x_j) = [\![\, P_j \,]\!] \;\mid\; j \in J\}\big)\{{}^{u}\!/\!{}_{x}\} \mid [\![\, b \,]\!]_u$
  $\gtrsim_{\mathrm{R}} \big((\boldsymbol{\nu} t)\,[\![\, v\{{}^{b}\!/\!{}_{x}\} \,]\!]_t \mid t!\mathrm{d}\,(\boldsymbol{\nu} a).a?\{\mathrm{l}_j(x_j) = [\![\, P_j\{{}^{b}\!/\!{}_{x}\} \,]\!] \;\mid\; j \in J\}\big)$
  Two cases : We either have $v = x$ or $v \neq x$.

  - $v = x$
    We have $[\![\, v \,]\!]_t = t \gg x$ so the whole expression is as follows :
    $(\boldsymbol{\nu} u)\,\big((\boldsymbol{\nu} t)\,t \gg u \mid t!\mathrm{d}\,(\boldsymbol{\nu} r).r?\{\mathrm{l}_j(x_j) = [\![\, P_j \,]\!]\{{}^{u}\!/\!{}_{x}\} \;\mid\; j \in J\} \mid [\![\, b \,]\!]_u\big)$
    $\gtrsim_{\mathrm{R}} (\boldsymbol{\nu} t)\,(t!\mathrm{d}\,(\boldsymbol{\nu} r).r?\{\mathrm{l}_j(x_j) = [\![\, P_j\{{}^{b}\!/\!{}_{x}\} \,]\!] \;\mid\; j \in J\} \mid [\![\, b \,]\!]_t)$.
    Let B be the process such that $[\![\, v \,]\!]_u \xrightarrow{u?\mathrm{d}\,\langle r\rangle} \equiv [\![\, v \,]\!]_u | B$. We have $b \notin \mathrm{fn}(B)$ and $[\![\, v \,]\!]_t \xrightarrow{t?\mathrm{d}\,\langle r\rangle} \equiv B$ (because t is linear)
    If we transmit r over t in the left hand side it gets into
    $(\boldsymbol{\nu} r)\,(\boldsymbol{\nu} u)\,\big(u!\mathrm{d}\,(\boldsymbol{\nu} s).s \gg r \,|\, r?\{\mathrm{l}_j(x_j) = [\![\, P_j \,]\!]\{{}^{u}\!/\!{}_{x}\} \;\mid\; j \in J\} \,|\, [\![\, b \,]\!]_u\big)$
    Removing the link :
    $\gtrsim_{\mathrm{R}} (\boldsymbol{\nu} u)\,\big(u!\mathrm{d}\,(\boldsymbol{\nu} r).r?\{\mathrm{l}_j(x_j) = [\![\, P_j \,]\!]\{{}^{u}\!/\!{}_{x}\} \;\mid\; j \in J\} \mid [\![\, b \,]\!]_u\big)$.
    Then transmitting r over u yields :
    $\xrightarrow{\tau} (\boldsymbol{\nu} u)\,\big(r?\{\mathrm{l}_j(x_j) = [\![\, P_j \,]\!]\{{}^{u}\!/\!{}_{x}\} \;\mid\; j \in J\} \mid [\![\, b \,]\!]_u \mid B\big)$
    Because u is only present in the input expression we can move u into it :
    $\sim_{\mathrm{R}} r?\{\mathrm{l}_j(x_j) = (\boldsymbol{\nu} u)\,([\![\, P_j \,]\!]\{{}^{u}\!/\!{}_{x}\} \mid [\![\, b \,]\!]_u) \;\mid\; j \in J\} \mid B$
    Concerning the right hand side :
    $(\boldsymbol{\nu} t)\,(t!\mathrm{d}\,(\boldsymbol{\nu} r).r?\{\mathrm{l}_j(x_j) = [\![\, P_j\{{}^{b}\!/\!{}_{x}\} \,]\!] \;\mid\; j \in J\} \mid [\![\, b \,]\!]_t)$
    Transmitting r over t yields :
    $\xrightarrow{\tau} (\boldsymbol{\nu} t)\,(r?\{\mathrm{l}_j(x_j) = [\![\, P_j\{{}^{b}\!/\!{}_{x}\} \,]\!] \;\mid\; j \in J\} \mid B)$
    because t has been consumed :
    $\equiv r?\{\mathrm{l}_j(x_j) = [\![\, P_j\{{}^{b}\!/\!{}_{x}\} \,]\!] \;\mid\; j \in J\} \mid B$

So to sum up we want to prove :
$r?\{\mathrm{l}_j(x_j) = (\boldsymbol{\nu} u)\,(\llbracket P_j \rrbracket\{^u\!/_x\} \mid \llbracket b \rrbracket_u) \mid j \in J\} \mid B \gtrsim_{\mathrm{R}}$
$r?\{\mathrm{l}_j(x_j) = \llbracket P_j\{^b\!/_x\} \rrbracket \mid j \in J\} \mid B$
which is true by induction hypothesis and $\gtrsim_{\mathrm{R}}$-congruence.

– $x \notin \mathrm{n}(v)$
In this case the u-related part (in the left hand side) can be moved inside the input expression and then we use congruence lemma :
$(\boldsymbol{\nu} t)\,\llbracket v \rrbracket_t \mid t!\mathrm{d}(\boldsymbol{\nu} a).a?\{\mathrm{l}_j(x_j) = (\boldsymbol{\nu} u)\,\llbracket P_j \rrbracket\{^u\!/_x\} \mid \llbracket b \rrbracket_u \mid j \in J\} \gtrsim_{\mathrm{R}} (\boldsymbol{\nu} t)\,\llbracket v \rrbracket_t \mid t!\mathrm{d}(\boldsymbol{\nu} a).a?\{\mathrm{l}_j(x_j) = \llbracket P_j\{^b\!/_x\} \rrbracket \mid j \in J\}$
is true if
$(\boldsymbol{\nu} u)\,\llbracket P_j \rrbracket\{^u\!/_x\} \mid \llbracket b \rrbracket_u \gtrsim_{\mathrm{R}} \llbracket P_j\{^b\!/_x\} \rrbracket$ which is true by induction hypothesis.

B.13 π_{a}^V to TyCO Operational Correspondence (5.2.6)

1. Let $P \rightarrowtail P'$. We then have (by definition of $\rightarrowtail$)
$P \equiv (\boldsymbol{\nu}\tilde{x})\,\mathsf{case}\,\mathrm{l}_k\langle w\rangle\,\mathsf{of}\,\{\mathrm{l}_j(x_j){=}P_j \mid j{\in}J\} \mid P_0$ with $k \in J$ and $P' \equiv P_k\{^w\!/_{x_k}\}$
So $\llbracket P \rrbracket \equiv (\boldsymbol{\nu}\tilde{x})\,(\boldsymbol{\nu} r)\,(\llbracket \mathrm{l}_k\langle w\rangle \rrbracket_r \mid r!\mathrm{d}(\boldsymbol{\nu} a).a?\{\mathrm{l}_j(x_j){=}\llbracket P_j \rrbracket \mid j{\in}J\}) | \llbracket P_0 \rrbracket$
$= (\boldsymbol{\nu}\tilde{x})\,(\boldsymbol{\nu} r)\,(r?\{\mathrm{d}(a) = a!\mathrm{l_k}(\boldsymbol{\nu} s).\llbracket w \rrbracket_s\} \mid$
$r!\mathrm{d}(\boldsymbol{\nu} a).a?\{\mathrm{l}_j(x_j){=}\llbracket P_j \rrbracket \mid j{\in}J\}) \mid \llbracket P_0 \rrbracket$
We can transmit a over r :
$(\boldsymbol{\nu}\tilde{x})\,(\boldsymbol{\nu} r)\,(\boldsymbol{\nu} a)\,(a!\mathrm{l_k}(\boldsymbol{\nu} s).\llbracket w \rrbracket_s\} \mid a?\{\mathrm{l}_j(x_j){=}\llbracket P_j \rrbracket \mid j{\in}J\} \mid \llbracket P_0 \rrbracket)$
Then s over a :
$(\boldsymbol{\nu}\tilde{x})\,(\boldsymbol{\nu} r)\,(\boldsymbol{\nu} a)\,(((\boldsymbol{\nu} s)\,\llbracket w \rrbracket_s \mid \llbracket P_k \rrbracket\{^s\!/_{x_k}\}) \mid \llbracket P_0 \rrbracket)$
By 5.2.5.2, $(\boldsymbol{\nu} s)\,\llbracket w \rrbracket_s \mid \llbracket P_k \rrbracket\{^s\!/_{x_k}\} \gtrsim_{\mathrm{R}} \llbracket P_k\{^w\!/_{x_k}\} \rrbracket$ and by 4.2.10 this can be applied to the whole process which is then $\gtrsim_{\mathrm{R}}$
$(\boldsymbol{\nu}\tilde{x})\,(\boldsymbol{\nu} r)\,(\boldsymbol{\nu} a)\,(\llbracket P_k\{^w\!/_{x_k}\} \rrbracket \mid \llbracket P_0 \rrbracket) \equiv$
$(\boldsymbol{\nu}\tilde{x})\,(\llbracket P_k\{^w\!/_{x_k}\} \rrbracket \mid \llbracket P_0 \rrbracket) \equiv \llbracket P' \rrbracket$
which is what we wanted.
Applying this many times we get : if $P \rightarrowtail^* \widetilde{P}'$ then $\llbracket P \rrbracket \Rightarrow\gtrsim_{\mathrm{R}} \llbracket \widetilde{P}' \rrbracket$.
2. We will now prove each entry of point 2 assuming that $P \xrightarrow{\mu} P'$ did not make use of (P-CASE), then apply point 1 and show that it preserves the results.
We can easily prove that if $P \xrightarrow{\mu} P'$ then there is $\widetilde{P}'$ such that $P \rightarrowtail^* \widetilde{P}' \xrightarrow{\mu} P'$ where the proof of $\widetilde{P}' \xrightarrow{\mu} P'$ doesn't make use of (P-CASE) (this is done by performing the case analysis required by $P \xrightarrow{\mu} P'$

in $P \rightarrowtail^* \widetilde{P'}$ itself, and then no further case analysis is required so (P-Case) isn't used in $\widetilde{P'} \xrightarrow{\mu} P'$)

(a) If $P \rightarrowtail^* \widetilde{P'} \xrightarrow{a!v} P'$ then the proof used (Out), and possibly some (Par$_1$) and (Res) so we have $\widetilde{P'} \equiv (\boldsymbol{\nu}\tilde{x})\, a!v \mid P_0$ with $(\{a\}, \mathrm{n}(v)) \cap \tilde{x} = \emptyset$.
Then, $[\![\, \widetilde{P'} \,]\!] \equiv (\boldsymbol{\nu}\tilde{x})\, a!\mathrm{c}(\boldsymbol{\nu}u).[\![\, v \,]\!]_u \mid [\![\, P_0 \,]\!] \xrightarrow{a!\mathrm{c}(\boldsymbol{\nu}u)} \equiv$ $[\![\, v \,]\!]_u \mid (\boldsymbol{\nu}\tilde{x})\, [\![\, P_0 \,]\!]$.
We also have $P' \equiv (\boldsymbol{\nu}\tilde{x})\, P_0$ so $[\![\, \widetilde{P'} \,]\!] \xrightarrow{a!\mathrm{c}(\boldsymbol{\nu}u)} \equiv ([\![\, v \,]\!]_u \mid [\![\, P' \,]\!])$.
So we have $[\![\, P \,]\!] \Rightarrow \gtrsim_{\mathrm{R}} \xrightarrow{a!\mathrm{c}(\boldsymbol{\nu}u)} \equiv ([\![\, v \,]\!]_u \mid [\![\, P' \,]\!])$ and then
$[\![\, P \,]\!] \xRightarrow{a!\mathrm{c}(\boldsymbol{\nu}u)} \gtrsim_{\mathrm{R}} ([\![\, v \,]\!]_u \mid [\![\, P' \,]\!])$

(b) $P \rightarrowtail^* \widetilde{P'} \xrightarrow{(\boldsymbol{\nu}b)\, a!v} P'$
Then $\widetilde{P'} \equiv (\boldsymbol{\nu}\tilde{x})\, a!v \mid P_0$ with $a \notin \tilde{x}$ and $b \in \tilde{x}$.
So, $[\![\, \widetilde{P'} \,]\!] \equiv (\boldsymbol{\nu}\tilde{x})\, a!\mathrm{c}(\boldsymbol{\nu}u).[\![\, v \,]\!]_u \mid [\![\, P_0 \,]\!] \xrightarrow{a!\mathrm{c}(\boldsymbol{\nu}u)} \equiv$ $[\![\, v \,]\!]_u \mid (\boldsymbol{\nu}\tilde{x})\, [\![\, P_0 \,]\!]$.
We also have $P' \equiv (\boldsymbol{\nu}\tilde{x} \setminus \{b\})\, P_0$ so
$[\![\, \widetilde{P'} \,]\!] \xrightarrow{a!\mathrm{c}(\boldsymbol{\nu}u)} \equiv (\boldsymbol{\nu}b)\, ([\![\, v \,]\!]_u \mid [\![\, P' \,]\!])$.
(We have to put back the restriction on b because the encoded side does not extrude the scope of plain names)
Similarly to free output, adding case-reductions we get
$[\![\, P \,]\!] \xRightarrow{a!\mathrm{c}(\boldsymbol{\nu}u)} \gtrsim_{\mathrm{R}} (\boldsymbol{\nu}b)\, ([\![\, v \,]\!]_u \mid [\![\, P' \,]\!])$

(c) $P \rightarrowtail^* \widetilde{P'} \xrightarrow{a?v} P'$ with $\mathrm{n}(v)$ fresh.
This case is more complex than other ones because receptive expansion would introduce a trigger name etc so we can't use point 2 of this proposition like in the other cases. We need to know dome details about the process structure.
The case reductions from P to $\widetilde{P'}$ will select subprocesses of P and apply some substitutions on them. Doing prior alpha renaming to make bound names fresh we can define P_0 to be the same as $\widetilde{P'}$ before the substitutions took place : $\widetilde{P'} = P_0\{^{\tilde{w}}/_{\tilde{r}}\}$.
On the encoded side we have $[\![\, P \,]\!] \Rightarrow (\boldsymbol{\nu}\tilde{u})\, ([\![\, \tilde{w} \,]\!]_{\tilde{u}} \mid [\![\, P_0 \,]\!]\{^{\tilde{u}}/_{\tilde{r}}\})$.
We will then assume $P_0 = a?(y).Q$, with $a \notin \tilde{r}$
So we have $[\![\, P \,]\!] \Rightarrow (\boldsymbol{\nu}\tilde{u})\, ([\![\, \tilde{w} \,]\!]_{\tilde{u}} \mid a?\{\mathrm{c}(y) {=} [\![\, Q \,]\!]\}\{^{\tilde{u}}/_{\tilde{r}}\})$
There may be additional restrictions and compositions but they are preserved during the whole process, just as in the cases handled

previously, so we dropped them for clarity. If we have a replicated input instead, then the proof is the same but just keeping the replicated input instead of consuming it after the input took place.
$\widetilde{P'} \xrightarrow{a?v} Q\{^{\tilde{w}}/_{\tilde{r}}\}\{^{v}/_{y}\}$ and, using (P-Case),
$P \xrightarrow{a?v} Q\{^{\tilde{w}}/_{\tilde{r}}\}\{^{v}/_{y}\} = P'$
And we have $[\![P]\!] \xRightarrow{a?\mathrm{c}\langle u\rangle} (\boldsymbol{\nu}\tilde{u})\,([\![\tilde{w}]\!]_{\tilde{u}} \mid [\![Q]\!]\{^{\tilde{u}}/_{\tilde{r}}\}\{^{u}/_{y}\}) = Q'$.
In the encoded side, to mimic an encoding of a transmission of v we introduce an encoding of v from u :
$(\boldsymbol{\nu}\tilde{u})\,(\boldsymbol{\nu}u)\,([\![v]\!]_u \mid [\![\tilde{w}]\!]_{\tilde{u}} \mid [\![Q]\!]\{^{\tilde{u}}/_{\tilde{r}}\}\{^{u}/_{y}\})$
Applying 5.2.5.1 as many times as required (once for each $\tilde{x}$ and once for r. Because all $\tilde{u}$ and u were chosen fresh there is no name collision problem), we have $(\boldsymbol{\nu}u)\,([\![v]\!]_u \mid Q') \gtrsim_{\mathrm{R}} [\![P']\!]$.

3. (a) Let $[\![P]\!] \xrightarrow{a!\mathrm{c}\,(\boldsymbol{\nu}u)} Q$.
The only output generated by the encoding is the $[\![a!v]\!]$ rule, and the only rules that will keep this output available are $[\![P_1|P_2]\!]$ and $[\![(\boldsymbol{\nu}a)\,P]\!]$. So we have $P \equiv (\boldsymbol{\nu}\tilde{x})\,a!v \mid R$.
Let $\mathrm{n}(v) = \{b\}$.
Note that we *can* have $b \in \tilde{x}$!
We have $[\![P]\!] \equiv (\boldsymbol{\nu}\tilde{x})\,(a!\mathrm{c}(\boldsymbol{\nu}u).[\![v]\!]_u \mid [\![R]\!])$.
$[\![P]\!] \xrightarrow{a!\mathrm{c}\,(\boldsymbol{\nu}u)} (\boldsymbol{\nu}\tilde{x})\,([\![v]\!]_u \mid [\![R]\!]) = Q$
First case, if $b \notin \tilde{x}$, we get
$P \xrightarrow{a!v}\equiv (\boldsymbol{\nu}\tilde{x})\,R = P'$ and $Q \equiv [\![P']\!] | [\![v]\!]_u$
Second case, if $b \in \tilde{x}$, we have to use rule (Open) and get
$P \xrightarrow{(\boldsymbol{\nu}b)\,a!v}\equiv (\boldsymbol{\nu}\tilde{x} \setminus b)\,R = P'$ and $Q \equiv (\boldsymbol{\nu}b)\,([\![P']\!] | [\![v]\!]_u)$.

(b) Let $[\![P]\!] \xrightarrow{a?\mathrm{c}\,\langle u\rangle} Q$.
The only rules of the encoding that generate an (observable) input are $[\![a?(x).P]\!]$ and $[\![a?^*(x).P]\!]$ and the input availability is preserved by parallel composition and restriction, just like the previous case.
We will handle the replicated case only, the single input one is similar. So we have :

- $P \equiv (\boldsymbol{\nu}\tilde{x})\,(a?^*(x).P_0 \mid R)$
- $P \xrightarrow{a?(v)} (\boldsymbol{\nu}\tilde{x})\,(a?^*(x).P_0 \mid P_0\{^{v}/_{x}\} \mid R) = P'$
- $[\![P]\!] \equiv (\boldsymbol{\nu}\tilde{x})\,(a?^*\{\mathrm{c}\,(x){=}[\![P_0]\!]\} \mid [\![R]\!])$

- $[\![P]\!] \xrightarrow{a?\mathrm{c}\langle u\rangle} \equiv (\boldsymbol{\nu}\tilde{x})\,(a?^*\{\mathrm{c}\,(x){=}[\![P_0]\!]\} \mid [\![P_0]\!]\{{}^{u}\!/\!_{x}\} \mid [\![R]\!]) = Q$

  Making v available on the encoded side we get
  $(\boldsymbol{\nu}u)\,([\![v]\!]_u \mid Q) \equiv$
  $(\boldsymbol{\nu}u)\,([\![v]\!]_u \mid (\boldsymbol{\nu}\tilde{x})\,(a?^*\{\mathrm{c}\,(x){=}[\![P_0]\!]\} \mid [\![P_0]\!]\{{}^{u}\!/\!_{x}\} \mid [\![R]\!])) \equiv$
  $(\boldsymbol{\nu}\tilde{x})\,(a?^*\{\mathrm{c}\,(x){=}[\![P_0]\!]\} \mid (\boldsymbol{\nu}u)\,([\![v]\!]_u \mid [\![P_0]\!]\{{}^{u}\!/\!_{x}\}) \mid [\![R]\!]) \gtrsim_{\mathrm{R}}$
  $(\boldsymbol{\nu}\tilde{x})\,(a?^*\{\mathrm{c}\,(x){=}[\![P_0]\!]\} \mid (\boldsymbol{\nu}u)\,([\![P_0\{{}^{v}\!/\!_{x}\}]\!] \mid [\![R]\!]) \mid\,) \equiv [\![P']\!]$

(c) Let $[\![P]\!] \xrightarrow{\tau} Q$.
This can happen because of an input/output pair transcripted from the source (π^V_{a}) process or a transmission of the r name over u in the process encoding a case expression.
So we have $P \equiv (\boldsymbol{\nu}\tilde{x})\,(P_i \mid P_o \mid R)$, where P_i does the input and P_o does the output.
$[\![P]\!] \equiv (\boldsymbol{\nu}\tilde{x})\,([\![P_i]\!] \mid [\![P_o]\!] \mid [\![R]\!])$ with
$[\![P_i]\!] \xrightarrow{a?\mathrm{c}\langle u\rangle} Q_i$ and $[\![P_o]\!] \xrightarrow{a!\mathrm{c}\,(\boldsymbol{\nu}u)} Q_o$ which gives $[\![P]\!] \xrightarrow{\tau}$ $(\boldsymbol{\nu}\tilde{x})\,(((\boldsymbol{\nu}u)\,Q_i \mid Q_o) \mid [\![R]\!]) = Q$
We now assume that the output of P_o is bound, as specified in the lemma
Using the previously proven points of this lemma we have
$P_o \xrightarrow{(\boldsymbol{\nu}b)\,a!v} P'_o$ and $P_i \xrightarrow{a?(v)} P'_i$, with
$Q_o \equiv (\boldsymbol{\nu}b)\,([\![P'_o]\!] | [\![v]\!]_u)$ and $(\boldsymbol{\nu}u)\,(Q_i | [\![v]\!]_u) \gtrsim_{\mathrm{R}} [\![P'_i]\!]$
So the transition on the source side is
$P \xrightarrow{\tau} \equiv (\boldsymbol{\nu}\tilde{x})\,(((\boldsymbol{\nu}b)\,P'_i \mid P'_o) \mid R) = P'$.
We now compare it with Q :
The relation between Q_o and P'_o gives
$Q \equiv (\boldsymbol{\nu}\tilde{x})\,(((\boldsymbol{\nu}u)\,Q_i \mid ((\boldsymbol{\nu}b)\,([\![P'_o]\!] | [\![v]\!]_u))) \mid [\![R]\!])$
We have $b \notin \mathrm{fn}(Q_i)$ so
$Q \equiv (\boldsymbol{\nu}\tilde{x})\,(((\boldsymbol{\nu}u, b)\,Q_i \mid [\![P'_o]\!] \mid [\![v]\!]_u) \mid [\![R]\!])$
Because $u \notin \mathrm{fn}(P'_o)$ (no encoded process has free receptive names)
$Q \equiv (\boldsymbol{\nu}\tilde{x})\,(((\boldsymbol{\nu}b)\,((\boldsymbol{\nu}u)\,Q_i \mid [\![v]\!]_u) \mid [\![P'_o]\!]) \mid [\![R]\!])$
Applying the Q_i - P'_i relation above :
$Q \gtrsim_{\mathrm{R}} (\boldsymbol{\nu}\tilde{x})\,(((\boldsymbol{\nu}b)\,[\![P'_i]\!] \mid [\![P'_o]\!]) \mid [\![R]\!]) = [\![P']\!]$
Which is completes the proof in the case of a communication on names of the original process.
We now prove the case where the transition is handling a case reduction.

We have $P \equiv (\boldsymbol{\nu}\tilde{x})\,\textsf{case}\,\mathrm{l}_k\langle w\rangle\,\textsf{of}\,\{\mathrm{l}_j(x_j){=}P_j \;\mid\; j{\in}J\} \mid R.\;(k \in J)$. The rest follows just as in point 1 of this proof.

B.14 π_{a}^{V}-TyCO Encoding Full Abstraction (5.2.7)

For shortness we will assume that we are working on processes for which 5.2.6 holds and not mention its side conditions.

We first prove that the set of pairs $([\![\, P_1 \,]\!];[\![\, P_2 \,]\!])$ with $P_1 \approx P_2$ is a receptive bisimilarity up to context.

All cases work the same way, starting from a transition on the TyCO side we first apply point 3 of the operational correspondence to get the behavior on the π_{a}^{V} side, then use the definition of weak bisimilarity to move from the $_1$ side to the $_2$ side and then apply point 2 of the operational correspondence to get the behavior of $[\![\, P_2 \,]\!]$. In all cases the resulting processes Q_1 and Q_2 are weakly receptive bisimilar respectively to $C[[\![\, P_1' \,]\!]]$ and $C[[\![\, P_2' \,]\!]]$, for some context.

Let $P_1 \approx P_2$ and $[\![\, P_1 \,]\!] \xrightarrow{\mu} Q_1$.

1. $\mu = a!\mathrm{c}(\boldsymbol{\nu}u)$:
   This might be for a bound output or a free output on the π_{a}^{V} side. We will assume it is bound output, and the proof in case of free output is the same just removing the restriction.
   By 5.2.6.3, there are b, v and P_1' such that $(\boldsymbol{\nu}b)\,([\![\, P_1' \,]\!] \mid [\![\, v \,]\!]_u) \equiv Q_1$ and $P_1 \xrightarrow{(\boldsymbol{\nu}b)\,a!v} P_1'$.
   By $\approx$, there is P_2' s.t. $P_2 \xRightarrow{(\boldsymbol{\nu}b)\,a!v} P_2'$ with $P_1' \approx P_2'$.
   By 5.2.6.2 (both input and τ cases), there is Q_2 s.t.
   $Q_2 \gtrsim_{\mathrm{R}} (\boldsymbol{\nu}b)\,([\![\, P_2' \,]\!] \mid [\![\, v \,]\!]_u)$.
2. $\mu = a?\mathrm{c}\langle u\rangle$:
   By 5.2.6.3, $P_1 \xrightarrow{a?(v)} P_1'$ with $(\boldsymbol{\nu}u)\,([\![\, v \,]\!]_u \mid Q_1) \gtrsim_{\mathrm{R}} [\![\, P_1' \,]\!]$.
   By $\approx$, $P_2 \xrightarrow{a?(v)} P_2'$ with $P_1' \approx P_2'$.
   By 5.2.6.2, $[\![\, P_2 \,]\!] \xRightarrow{a?\mathrm{c}\langle u\rangle} Q_2$ with $(\boldsymbol{\nu}u)\,([\![\, v \,]\!]_u \mid [\![\, Q_2 \,]\!]) \gtrsim_{\mathrm{R}} [\![\, P_2' \,]\!]$.
   Taking $v = t$ (t being some fresh trigger name), we meet the requirements of weak receptive bisimilarity (using identity instead of $\Rightarrow$ in clause (b)).
3. $\mu = \tau$:

By 5.2.6.3, we either have $P_1 \xrightarrow{\tau} P_1'$ with $Q_1 \gtrsim_{\mathrm{R}} [\![P_1']\!]$ or $P_1 \rightarrowtail P_1'$, $Q_1 \xrightarrow{\tau} Q_1'$ with $Q_1' \gtrsim_{\mathrm{R}} [\![P_1']\!]$.
In first case, weak bisimilarity gives $P_2 \Rightarrow P_2'$ with $P_1' \approx P_2'$. In second case we have $P_1' \approx P_2$ (because $A \rightarrowtail B$ implies $A \sim B$, as a consequence of lemma 2.2.5). In that second case, let $P_2' = P_2$.
So, by 5.2.6.2, $[\![P_2]\!] \xRightarrow{Q_2}$ with $Q_2 \gtrsim_{\mathrm{R}} [\![P_2']\!]$

We will now prove the other direction.

On the encoded side the values aren't directly transmitted, fresh references u are used instead. This means that when a value is sent, no matter what it is and if it is bound or not, the transition on the encoded side will be the same.

However the contents of this value can then be accessed by sending a d message to u, and the result of this must be preserved by the $[\![P_i]\!]$ weak receptive bisimilarity.

Lemma B.14.1. *Let* $P_1 = (\boldsymbol{\nu}\tilde{x}_1)\,([\![v_1]\!]_u \mid R_1) \approx_{\mathrm{R}} (\boldsymbol{\nu}\tilde{x}_2)\,([\![v_2]\!]_u \mid R_2) = P_2$ *(*$u \in \mathbf{N}_{\mathrm{u}}$*)*

Let $\mathrm{n}(v_i) = \{b_i\}$*.*

We then have $v_1 = v_2$ *(so* $b_1 = b_2$*) and* $b_1 \in \tilde{x}_1$ *iff* $b_2 \in \tilde{x}_2$*.*

Proof. Three cases.

1. $v_1 = b_1$ and $b_1 \notin \tilde{x}_1$, then
   We send a query to u with a fresh t :
   $P_1 = (\boldsymbol{\nu}\tilde{x}_1)\,(u \gg b_1 \mid R_1) \xrightarrow{u?\mathrm{d}\,\langle t\rangle}$
   The answer is an output at b_1 with a new name t' :
   $(\boldsymbol{\nu}\tilde{x}_1)\,(u \gg b_1 \mid b_1!\mathrm{d}(\boldsymbol{\nu}r).r \gg t \mid R_1) \xrightarrow{b_1!\mathrm{d}\,(\boldsymbol{\nu}r)}$
   We check that this name is linked to "our" name (t) sending a query to it :
   $(\boldsymbol{\nu}\tilde{x}_1)\,(u \gg b_1 \mid r \gg t \mid R_1) \xrightarrow{r?\mathrm{l}\langle t'\rangle}$
   As an answer we get an output at our fresh t which is what we required.
   $(\boldsymbol{\nu}\tilde{x}_1)\,(u \gg b_1 \mid t!\mathrm{l}(\boldsymbol{\nu}r').r' \gg t' \mid R_1) \xrightarrow{t!\mathrm{l}(\boldsymbol{\nu}r')}$.
2. if $v_1 = \mathrm{l}\langle w\rangle$.
   We send a query to u with a fresh t :
   $P_1 = (\boldsymbol{\nu}\tilde{x}_1)\,(u?\{\mathrm{d}(a) = a!\mathrm{l}(\boldsymbol{\nu}s).[\![w]\!]_s\} \mid R_1) \xrightarrow{u?\mathrm{d}\,\langle t\rangle}$
   As answer there is an output at t with a reference to the embedded value w :

$(\boldsymbol{\nu}\tilde{x}_1)\,(u?\{\mathrm{d}(a) = a!\mathrm{l}(\boldsymbol{\nu}s).[\![\, w \,]\!]_s\} \mid t!\mathrm{l}(\boldsymbol{\nu}s).[\![\, w \,]\!]_s\} \mid R_1) \xrightarrow{t!\mathrm{l}(\boldsymbol{\nu}s)}$
The same protocol can be ran to analyse the w value :
$(\boldsymbol{\nu}\tilde{x}_1)\,(u?\{\mathrm{d}(a) = a!\mathrm{l}(\boldsymbol{\nu}s).[\![\, w \,]\!]_s\} \mid [\![\, w \,]\!]_s \mid R_1) \equiv$
$(\boldsymbol{\nu}\tilde{x}_1)\,([\![\, w \,]\!]_s \mid ([\![\, v \,]\!]_u|R_1))$

3. v_1 is a simple name that is part of $\tilde{x}_1$, then we can't get an answer to a query at u ($\xrightarrow{u?\mathrm{d}\,\langle t\rangle}$), ie no transition involving t in output can occur after this one.

If we have $P_1 \approx_{\mathrm{R}} P_2$ then P_2 must have the same sequence of transitions, so, because all three cases can always be distinguished from each other, we must have the same values v_1 v_2, up to alpha-renaming in case b_1 is bound in $\tilde{x}_1$ (in which case b_2 must be bound as well otherwise P_2 would respond in case 1 instead of case 3)

□

We assume $[\![\, P_1 \,]\!] \approx_{\mathrm{R}} [\![\, P_2 \,]\!]$. Let $P_1 \xrightarrow{\mu} P_1'$.

1. $\mu = a!v$ or $\mu = (\boldsymbol{\nu}b)\,a!v$
   By 5.2.6.2, we have $[\![\, P_1 \,]\!] \xRightarrow{a!\mathrm{c}\,(\boldsymbol{\nu}u)} Q_1$ with $Q_1 \gtrsim_{\mathrm{R}} (\boldsymbol{\nu}b)\,([\![\, P_1' \,]\!] \mid [\![\, v \,]\!]_u)$ or the same without the restriction.
   By $\approx_{\mathrm{R}}$, we have $[\![\, P_2 \,]\!] \xRightarrow{a!\mathrm{c}\,(\boldsymbol{\nu}u)} Q_2$ with $Q_2 \approx_{\mathrm{R}} Q_1$.
   The only observable output of an encoded process is produced by the $[\![\, a!v \,]\!]$ rule, and it is kept observable by parallel composition and restriction so we have
   $[\![\, P_2 \,]\!] \equiv (\boldsymbol{\nu}\tilde{x})\,(a!\mathrm{c}(\boldsymbol{\nu}u).[\![\, w \,]\!]_u \mid [\![\, R \,]\!]) \xrightarrow{a!\mathrm{c}\,(\boldsymbol{\nu}u)}$

$$(\boldsymbol{\nu}\tilde{x})\,([\![\, w \,]\!]_u \mid R) \approx_{\mathrm{R}} (\boldsymbol{\nu}b)\,([\![\, v \,]\!]_u \mid [\![\, P_1' \,]\!])(\text{or without the } (\boldsymbol{\nu}b)\,) \qquad (3)$$

   Applying the above lemma we get $\mathrm{n}(w) \subseteq \tilde{x}$ if and only if the b restriction is there, and $v = w$ up to alpha-renaming in case it is restricted.
   We have $P_2 \equiv (\boldsymbol{\nu}\tilde{x})\,(a!w \mid R) \xrightarrow{\mu} (\boldsymbol{\nu}\tilde{x} \setminus \mathrm{n}(w))\,R = P_2'$
   Removing the v encoding and if needed the restricted name from both sides of (3) we get $[\![\, P_2' \,]\!] = (\boldsymbol{\nu}\tilde{x} \setminus \mathrm{n}(w))\,R \approx_{\mathrm{R}} [\![\, P_1' \,]\!]$
2. $\mu = a?(v)$
   By 5.2.6.2, $[\![\, P_1 \,]\!] \xRightarrow{a?\mathrm{c}\,\langle u\rangle} Q_1$ with $(\boldsymbol{\nu}u)\,([\![\, v \,]\!]_u \mid Q_1) \gtrsim_{\mathrm{R}} [\![\, P_1' \,]\!]$.
   By $\approx_{\mathrm{R}}$, $[\![\, P_2 \,]\!] \xRightarrow{a?\mathrm{c}\,\langle u\rangle} Q_2$, $(\boldsymbol{\nu}u)\,(u \gg t \mid Q_2) \Rightarrow (\boldsymbol{\nu}u)\,(u \gg t \mid Q_2')$ (t fresh) with $(\boldsymbol{\nu}u)\,(u \gg t \mid Q_1) \approx_{\mathrm{R}} (\boldsymbol{\nu}u)\,(u \gg t \mid Q_2')$.

We can replace $u \gg t$ by $(\boldsymbol{\nu} t)\,(u \gg t \mid [\![v]\!]_t)$ and still have the same transitions and the latter, by 5.2.5.1, $\gtrsim_{\mathrm{R}} [\![v]\!]_u$.
This gives $(\boldsymbol{\nu} u)\,([\![v]\!]_u \mid Q_2) \Rightarrow (\boldsymbol{\nu} u)\,([\![v]\!]_u \mid Q_2'')$ (we can have $Q_2' \neq Q_2''$) with $(\boldsymbol{\nu} u)\,([\![v]\!]_u \mid Q_1) \approx_{\mathrm{R}} (\boldsymbol{\nu} u)\,([\![v]\!]_u \mid Q_2'')$.
5.2.6.3 gives $P_2 \xRightarrow{a?(v)} P_2'$ with $(\boldsymbol{\nu} u)\,([\![v]\!]_u \mid Q_2) \gtrsim_{\mathrm{R}} [\![P_2']\!]$.
The additional transitions $(\boldsymbol{\nu} u)\,([\![v]\!]_u \mid Q_2) \Rightarrow (\boldsymbol{\nu} u)\,([\![v]\!]_u \mid Q_2'')$ have to be reflected :
$[\![P_2']\!] \Rightarrow \widetilde{P}_2'' \approx_{\mathrm{R}} (\boldsymbol{\nu} u)\,([\![v]\!]_u \mid Q_2'')$ and operational correspondence gives $P_2' \Rightarrow P_2''$ and $[\![P_2'']\!] \approx_{\mathrm{R}} \widetilde{P}_2''$.
As a summary we have $P_2 \xRightarrow{a?(v)} P_2'' \approx_{\mathrm{R}} \widetilde{P}_2'' \approx_{\mathrm{R}} (\boldsymbol{\nu} u)\,([\![v]\!]_u \mid Q_2'') \approx_{\mathrm{R}} (\boldsymbol{\nu} u)\,([\![v]\!]_u \mid Q_1) \gtrsim_{\mathrm{R}} [\![P_1']\!]$

3. $\mu = \tau$
5.2.6.1 can be weakened like this : if $P \rightarrowtail P'$ then $[\![P]\!] \gtrsim_{\mathrm{R}} [\![P']\!]$ (that is because the two τ transitions used to process the case reduction work internally and are deterministic)
So the τ case of 5.2.6.3 can be weakened like this : $\forall P, Q. [\![P]\!] \Rightarrow Q$ implies there is P' with $P \Rightarrow P'$ and $Q \gtrsim_{\mathrm{R}} [\![P']\!]$.
By 5.2.6.2, $[\![P_1]\!] \xRightarrow{\tau} Q_1$ with $Q_1 \gtrsim_{\mathrm{R}} [\![P_1']\!]$.
By $\approx_{\mathrm{R}}$, $[\![P_2]\!] \Rightarrow Q_2$ with $Q_1 \approx_{\mathrm{R}} Q_2$.
Using the weakened τ operational correspondence we get
$P_2 \Rightarrow P_2'$ with $Q_2 \gtrsim_{\mathrm{R}} [\![P_2']\!]$.
So $[\![P_1']\!] \approx_{\mathrm{R}} [\![P_2']\!]$.

References

[Bor98] M. Boreale. On the Expressiveness of Internal Mobility in Name-Passing Calculi. *Theoretical Computer Science*, 195(2):205–226, 1998. An extended abstract appeared in *Proceedings of CONCUR '96*, LNCS 1119: 163–178.

[Mer00] M. Merro. *Locality in the π-calculus and applications to distributed objects*. PhD thesis, Ecole des Mines, France, October 2000.

[MPW92] R. Milner, J. Parrow and D. Walker. A Calculus of Mobile Processes, Part I/II. *Information and Computation*, 100:1–77, Sept. 1992.

[MS98] M. Merro and D. Sangiorgi. On Asynchrony in Name-Passing Calculi. In K. G. Larsen, S. Skyum and G. Winskel, eds, *Proceedings of ICALP '98*, volume 1443 of *LNCS*, pages 856–867. Springer, July 1998.

[Nes00] U. Nestmann. What Is a 'Good' Encoding of Guarded Choice? *Information and Computation*, 156:287–319, 2000. An extended abstract appeared in the *Proceedings of EXPRESS '97*, volume 7 of *ENTCS*.

[NP00] U. Nestmann and B. C. Pierce. Decoding Choice Encodings. *Information and Computation*, October/November 2000. To appear. Available as report BRICS-RS-99-42, Universities of Aalborg and Århus, Denmark, 1999.

[Pal97] C. Palamidessi. Comparing the Expressive Power of the Synchronous and the Asynchronous π-calculus. In *Proceedings of POPL '97*, pages 256–265. ACM, Jan. 1997.

[San93] D. Sangiorgi. From π-calculus to Higher-Order π-calculus — and back. In M.-C. Gaudel and J.-P. Jouannaud, eds, *Proceedings of TAPSOFT '93*, volume 668 of *LNCS*, pages 151–166. Springer, 1993.

[San96] D. Sangiorgi. π-Calculus, Internal Mobility and Agent-Passing Calculi. *Theoretical Computer Science*, 167(1,2):235–274, 1996. Also as Rapport de Recherche RR-2539, INRIA Sophia-Antipolis, 1995. Extracts of parts of the material contained in this paper can be found in *Proceedings of TAPSOFT '95* and *ICALP '95*.

[San98] D. Sangiorgi. An Interpretation of Typed Objects into Typed π-Calculus. *Information and Computation*, 143(1):34–73, 1998. Earlier version published as Rapport de Recherche RR-3000, INRIA Sophia-Antipolis, August 1996, and presented at FOOL 3.

[San99a] D. Sangiorgi. The Name Discipline of Uniform Receptiveness. *Theoretical Computer Science*, 221(1–2):457–493, 1999. An abstract appeared in the *Proceedings of ICALP '97*, LNCS 1256, pages 303–313.

[San99b] D. Sangiorgi. The Typed π-Calculus at work: A Proof of Jones's Parallelisation Theorem on Concurrent Objects. *Theory and Practice of Object-Oriented Systems*, 5(1), 1999. An early version was included in the *Informal proceedings of FOOL 4*, January 1997.

[Vas94] V. T. Vasconcelos. *A process-calculus approach to typed concurrent objects*. PhD thesis, Keio University, 1994.