Proof-relevant pi-calculus

Proof-relevant π-calculus

Roly Perera
University of Glasgow
Glasgow, UK
roly.perera@glasgow.ac.uk

James Cheney
University of Edinburgh
Edinburgh, UK
jcheney@inf.ed.ac.uk

Formalising the π-calculus is an illuminating test of the expressiveness of logical frameworks and mechanised metatheory systems, because of the presence of name binding, labelled transitions with name extrusion, bisimulation, and structural congruence. Formalisations have been undertaken in a variety of systems, primarily focusing on well-studied (and challenging) properties such as the theory of process bisimulation. We present a formalisation in Agda that instead explores the theory of concurrent transitions, residuation, and causal equivalence of traces, which has not previously been formalised for the π-calculus. Our formalisation employs de Bruijn indices and dependently-typed syntax, and aligns the "proved transitions" proposed by Boudol and Castellani in the context of CCS with the proof terms naturally present in Agda's representation of the labelled transition relation. Our main contributions are proofs of the "diamond lemma" for residuation of concurrent transitions and a formal definition of equivalence of traces up to permutation of transitions.

1 Introduction

The π-calculus [18, 19] is an expressive model of concurrent and mobile processes. It has been investigated extensively and many variations, extensions and refinements have been proposed, including the asynchronous, polyadic, and applied π-calculus (among many others). The π-calculus has also attracted considerable attention from the logical frameworks and meta-languages community, and formalisations of its syntax and semantics have been performed using most of the extant mechanised metatheory techniques, including (among others) Coq [13, 12, 15], Nominal Isabelle [2], Abella [1] (building on Miller and Tiu [26]), CLF [6], and Agda [21]. These formalisations have overcome challenges that tested the limits of these systems (at least at the time), particularly relating to the encoding of name binding, scope extrusion and structural congruence. Indeed, some early formalisations motivated or led to important contributions to the understanding of these issues in different systems, such as the Theory of Contexts, or CLF's support for monadic encapsulation of concurrent executions.

Prior formalisations have typically considered the syntax, semantics (usually via a variation on labelled transitions), and bisimulation theory of the π-calculus. However, as indicated above, while these aspects of the π-calculus are essential, they only scratch the surface of the properties that could be investigated. Most of these developments have been carried out using informal paper proofs, and formalising them may reveal challenges or motivate further research on logical frameworks.

One interesting aspect of the π-calculus that has not been formally investigated, and remains to some extent ill-understood informally, is its theory of *causal equivalence*. Two transitions t_1, t_2 that can be taken from a process term p are said to be *concurrent* ($t_1 \frown t_2$) if they could be performed "in either order" — that is, if after performing t_1, there is a natural way to transform the other transition t_2 so that its effect is performed on the result of t_1, and vice versa. The translation of the second transition is said to be the *residual* of t_2 after t_1, written t_2/t_1. The key property of this operation, called the "diamond lemma", is that the two residuals t_1/t_2 and t_2/t_1 of transitions $t_1 \frown t_2$ result in the same process. Finally,

Submitted to:
LFMTP 2015

permutation of concurrent transitions induces a *causal equivalence* relation on pairs of traces. This is the standard notion of permutation-equivalence from the theory of traces over concurrent alphabets [17].

Our interest in this area stems from previous work on provenance, slicing and explanation (e.g. [22]), which we wish to adapt to concurrent settings. Ultimately, we would like to formalise the relationship between informal "provenance graphs" often used informally to represent causal relationships [7] and the semantics of concurrent languages and traces. The π-calculus is a natural starting point for this study. We wish to understand how to represent, manipulate, and reason about π-calculus execution traces safely: that is, respecting well-formedness and causality.

In classical treatments, starting with Lévy [16], a transition is usually considered to be a triple (e, t, e') where e and e' are the terms and t is some information about the step performed. Boudol and Castellani [4] introduced the *proved transitions* approach for CCS in which the labels of transitions are enriched with more information about the transition performed. Boreale and Sangiorgi [3] and Degano and Priami [11] developed theories of causal equivalence for the π-calculus, building indirectly on the proved transition approach; Danos and Krivine [10] and Cristescu, Krivine and Varacca [8] developed notions of causality in the context of reversible CCS and π-calculus respectively. However, there does not appear to be a consensus about the correct definition of causal equivalence for the π-calculus. For example, Cristescu et al. [8] write "[in] the absence of an indisputable definition of permutation equivalence for [labelled transition system] semantics of the π-calculus it is hard to assert the correctness of one definition over another." In their work on reversible π-calculus, they noted that some previous treatments of causality in the π-calculus did not allow permuting transitions within the scope of a ν-binder, and showed how their approach would allow this. Moreover, none of the above approaches has been formalised.

In this paper, we report on a new formalisation of the π-calculus carried out in the dependently-typed programming language Agda [20]. Our main contributions include formalisations of concurrency, residuation, the diamond lemma, and causal equivalence. We do not attempt to formalise the above approaches directly, any one of which seems to be a formidable challenge. Instead, we have chosen to adapt the ideas of Boudol and Castellani to the π-calculus as directly as we can, guided by the hypothesis that their notion of *proved transitions* can be aligned with the *proof terms* for transition steps that arise naturally in a constructive setting. For example, we define the concurrency relation on (compatibly-typed) transition proof terms, and we define residuation as a total function taking two transitions along with a proof that the transitions are concurrent, rather than having to deal with a partial operation.

Our formalisation employs de Bruijn indices [5], an approach with well-known strengths and weaknesses compared, for example, to higher-order or nominal abstract syntax techniques employed in existing formalisations. For convenience, we employ a restricted form of structural congruence called *braiding congruence*, and we have not formalised as many of the classical results on the π-calculus as others have, but we do not believe there are major obstacles to filling these gaps. To the best of our knowledge, ours is the first mechanised proof of the diamond lemma for any process calculus.

The rest of the paper is organised as follows. §2 presents our variant of the (synchronous) π-calculus, including syntax, renamings, transitions and braiding congruence. §3 presents our definitions of concurrency and residuation for transitions, and discusses the diamond lemma. §4 presents our definition of causal equivalence. §5 discusses related work in greater detail and §6 concludes and discusses prospects for future work. Appendix A summarises the Agda module structure; the source code can be found at `https://github.com/rolyp/proof-relevant-pi`, release 0.1. Appendix B contains graphical proof-sketches for some lemmas, and Appendix C some further examples of residuation.

2 Synchronous π-calculus

We present our formalisation in the setting of a first-order, synchronous, monadic π-calculus with recursion and internal choice, using a labelled transition semantics. The syntax of the calculus is conventional (using de Bruijn indices) and is given below.

Name	x, y, z	$::=$	$0 \mid 1 \mid \cdots$		Process	P, Q, R, S	$::=$	$\mathbf{0}$	inactive
Action	a	$::=$	$\underline{x}$	input				$\underline{x}.P$	input
			$\overline{x}\langle y \rangle$	output				$\overline{x}\langle y \rangle.P$	output
			$\overline{x}$	bound output				$P + Q$	choice
			τ	silent				$P \mid Q$	parallel
								νP	restriction
								$!P$	replication

Names are ranged over by x, y and z. An input action is written $\underline{x}$. Output actions are written $\overline{x}\langle y \rangle$ if y is in scope and $\overline{x}$ if the action represents the output of a name whose scope is extruding, in which case we say the action is a *bound* output. Bound outputs do not appear in user code but arise during execution.

To illustrate, the conventional π-calculus term $(\nu x)\, x(z).\overline{y}\langle z \rangle.0 \mid \overline{x}\langle c \rangle.0$ would be represented using de Bruijn indices as $\nu(\underline{0}.\overline{n+1}\langle 0 \rangle.\mathbf{0} \mid \overline{0}\langle m+1 \rangle.\mathbf{0})$, provided that y and c are associated with indices n and m. Here, the first 0 represents the bound variable x, the second 0 the bound variable z, and the third refers to x again. Note that the symbol $\mathbf{0}$ denotes the inactive process term, not a de Bruijn index.

Let Γ and Δ range over *contexts*, which are finite initial segments of the natural numbers. The function which extends a context with a new element is written as a postfix $\cdot + 1$. A context Γ *closes* P if Γ contains the free variables of P. We denote by $\mathsf{Proc}\,\Gamma$ the set of processes closed by Γ, as defined below. We write $\Gamma \vdash P$ to mean $P \in \mathsf{Proc}\,\Gamma$. Similarly, actions are well-formed only in closing contexts; we write $a : \mathsf{Action}\,\Gamma$ to mean that Γ is closing for a, as defined below.

$\Gamma \vdash P$

$$\frac{}{\Gamma \vdash \mathbf{0}} \qquad \frac{\Gamma + 1 \vdash P}{\Gamma \vdash \underline{x}.P}\ x \in \Gamma \qquad \frac{\Gamma \vdash P}{\Gamma \vdash \overline{x}\langle y \rangle.P}\ x, y \in \Gamma \qquad \frac{\Gamma \vdash P \qquad \Gamma \vdash Q}{\Gamma \vdash P + Q} \qquad \frac{\Gamma \vdash P \qquad \Gamma \vdash Q}{\Gamma \vdash P \mid Q}$$

$$\frac{\Gamma + 1 \vdash P}{\Gamma \vdash \nu P} \qquad \frac{\Gamma \vdash P}{\Gamma \vdash !P}$$

$a : \mathsf{Action}\,\Gamma$

$$\frac{}{\underline{x} : \mathsf{Action}\,\Gamma}\ x \in \Gamma \qquad \frac{}{\overline{x} : \mathsf{Action}\,\Gamma}\ x \in \Gamma \qquad \frac{}{\overline{x}\langle y \rangle : \mathsf{Action}\,\Gamma}\ x, y \in \Gamma \qquad \frac{}{\tau : \mathsf{Action}\,\Gamma}$$

To specify the labelled transition semantics, it is convenient to distinguish *bound* actions b from non-bound actions c. A bound action $b : \mathsf{Action}\,\Gamma$ is of the form $\underline{x}$ or $\overline{x}$, and shifts a process from Γ to a target context $\Gamma + 1$, freeing the index 0. A non-bound action $c : \mathsf{Action}\,\Gamma$ is of the form $\overline{x}\langle y \rangle$ or τ, and has a target context which is also Γ. Meta-variable a ranges over all actions, bound and non-bound.

2.1 Renamings

A de Bruijn indices formulation of π-calculus makes extensive use of renamings. A *renaming* $\rho : \Gamma \longrightarrow \Delta$ is any function (injective or otherwise) from Γ to Δ. The labelled transition semantics makes use of the lifting of the successor function $\cdot + 1$ on natural numbers to renamings, which we call push to avoid confusion with the $\cdot + 1$ operation on contexts; pop y which undoes the effect of push, replacing 0 by y; and swap, which transposes the roles of 0 and 1. This de Bruijn treatment of π-calculus is similar to that of Hirschkoff's asynchronous μs calculus [14], except that we give a late rather than early semantics; other differences are discussed in §5 below.

$\mathsf{push}_\Gamma : \Gamma \longrightarrow \Gamma + 1$

$$\mathsf{push}\ x = x + 1$$

$\mathsf{pop}_\Gamma : \Gamma \longrightarrow \Gamma + 1 \longrightarrow \Gamma$

$$\mathsf{pop}\ y\ 0 = y$$
$$\mathsf{pop}\ y\ (x+1) = x$$

$\mathsf{swap}_\Gamma : \Gamma + 2 \longrightarrow \Gamma + 2$

$$\mathsf{swap}\ 0 = 1$$
$$\mathsf{swap}\ 1 = 0$$
$$\mathsf{swap}\ (x+2) = x + 2$$

The Γ subscripts that appear on push_Γ, $\mathsf{pop}_\Gamma\ x$ and swap_Γ are shown in grey to indicate that they may be omitted when their value is obvious or irrelevant; this is a convention we use throughout the paper.

2.1.1 Lifting renamings to processes and actions

The functorial extension $\rho^* : \mathsf{Proc}\ \Gamma \longrightarrow \mathsf{Proc}\ \Delta$ of a renaming $\rho : \Gamma \longrightarrow \Delta$ to processes is defined in the usual way. Renaming under a binder utilises the action of $\cdot + 1$ on renamings, which is also functorial. Syntactically, ρ^* binds tighter than any process constructor.

$\cdot^* : (\Gamma \longrightarrow \Delta) \longrightarrow \mathsf{Proc}\ \Gamma \longrightarrow \mathsf{Proc}\ \Delta$

$$\rho^* \mathbf{0} = \mathbf{0}$$
$$\rho^* (\underline{x}.P) = \underline{\rho x}.(\rho+1)^* P$$
$$\rho^* (\overline{x}\langle y \rangle.P) = \overline{\rho x}\langle \rho y \rangle.\rho^* P$$
$$\rho^* (P + Q) = \rho^* P + \rho^* Q$$
$$\rho^* (P \mid Q) = \rho^* P \mid \rho^* Q$$
$$\rho^* (\nu P) = \nu(\rho+1)^* P$$
$$\rho^* (!P) = !\rho^* P$$

$\cdot^* : (\Gamma \longrightarrow \Delta) \longrightarrow \mathsf{Action}\ \Gamma \longrightarrow \mathsf{Action}\ \Delta$

$$\rho^*\ \underline{x} = \underline{\rho x}$$
$$\rho^*\ \overline{x} = \overline{\rho x}$$
$$\rho^*\ \tau = \tau$$
$$\rho^*\ \overline{x}\langle y \rangle = \overline{\rho x}\langle \rho y \rangle$$

$\cdot + 1 : (\Gamma \longrightarrow \Delta) \longrightarrow \Gamma + 1 \longrightarrow \Delta + 1$

$$(\rho + 1)\ 0 = 0$$
$$(\rho + 1)\ (x+1) = \rho x + 1$$

2.1.2 Properties of renamings

Several equational properties of renamings are used throughout the development; here we present the ones mentioned elsewhere in the paper. Diagrammatic versions of the lemmas, along with string diagrams that offer a graphical intuition for why the lemmas hold, are given in Appendix B.

Lemma 1. $pop\ x \circ push = id$

Freeing the index 0 and then immediately substituting x for it is a no-op.

Lemma 2. $pop\ 0 \circ push + 1 = id$

Lemma 3. $swap + 1 \circ swap \circ swap + 1 = swap \circ swap + 1 \circ swap$

The above are two equivalent ways of swapping indices 0 and 2.

Lemma 4. $pop\ 0 \circ swap = pop\ 0$

Lemma 5. $swap \circ push + 1 = push$, $swap \circ push = push + 1$

Lemma 6. $push \circ \rho = \rho + 1 \circ push$

Lemma 7. $\rho \circ pop\ x = pop\ \rho x \circ \rho + 1$

Lemma 8. $swap \circ \rho + 2 = \rho + 2 \circ swap$

These last two lemmas assert various naturality properties of push, pop x and swap.

2.2 Labelled transition semantics

An important feature of our semantics is that each transition rule has an explicit constructor name. This allow derivations to be written in a compact, expression-like form, similar to the *proven transitions* used by Boudol and Castellani to define notions of concurrency and residuation for CCS [4]. However, rather than giving an additional inductive definition describing the structure of a "proof" that $P \xrightarrow{a} R$, we simply treat the inductive definition of $\longrightarrow$ as a data type. This is a natural approach in a dependently-typed setting.

The rule names are summarised below, and have been chosen to reflect, where possible, the structure of the process triggering the rule. The corresponding relation $P \xrightarrow{a} R$ is defined in Figure 1, for any process $\Gamma \vdash P$, any $a : \mathrm{Action}\ \Gamma$ with target $\Delta \in \{\Gamma, \Gamma + 1\}$, and any $\Delta \vdash R$.

Transition	E, F	$::=$	$\underline{x}.P$	input on x
			$\overline{x}\langle y \rangle.P$	output y on x
			$E + Q \quad P + F$	choose left or right branch
			$E\ {}^{a}\vert\ Q \quad P \vert^{a}\ F$	propagate a through parallel composition on the left or right
			$E \vert^{\tau}_{y} F \quad E\ {}^{\tau}_{y}\vert\ F$	rendezvous (receiving y on the left or right)
			$\overline{\nu} E$	initiate name extrusion
			$E \vert^{\tau}_{\nu} F \quad E\ {}^{\tau}_{\nu}\vert\ F$	extrusion rendezvous (receiving 0 on the left or right)
			$\nu^{a} E$	propagate a through binder
			$!E$	replicate

The constructor name for each rule is shown to the left of the rule. There is an argument position, indicated by $\cdot$, for each premise of the rule. Note that there are two forms of the transition constructors ${}^{a}\vert\cdot$ and $\nu^{a}\cdot$ distinguished by whether they are indexed by a bound action b or by a non-bound action c. Moreover there are additional (but symmetric) rules of the form $P + \cdot$, $P \vert^{b} \cdot$ and $P \vert^{b} \cdot$ where the sub-transition occurs on the opposite side of the operator, and similarly ${}^{\tau}_{\nu}\vert\cdot$ and ${}^{\tau}_{y}\vert\cdot$ rules in which the positions of sender and receiver are transposed. These are all straightforward variants of the rules shown, and are omitted from Figure 1 for brevity. Meta-variables E and F range over transition derivations; if $E : P \xrightarrow{a} R$ then $\mathrm{src}(E)$ denotes P and $\mathrm{tgt}(E)$ denotes R.

Although a de Bruijn formulation of pi calculus requires a certain amount of housekeeping, one pleasing consequence is that the usual side-conditions associated with the π-calculus transition rules are either subsumed by syntactic constraints on actions, or "operationalised" using the renamings above. In particular:

$P \xrightarrow{a} R$

$$\underline{x}.P \;\frac{}{\underline{x}.P \xrightarrow{\underline{x}} P} \qquad \overline{x}\langle y\rangle.P \;\frac{}{\overline{x}\langle y\rangle.P \xrightarrow{\overline{x}\langle y\rangle} P} \qquad \cdot + Q \;\frac{P \xrightarrow{a} R}{P+Q \xrightarrow{a} R}$$

$$\cdot^{c}|\, Q \;\frac{P \xrightarrow{c} R}{P \mid Q \xrightarrow{c} R \mid Q} \qquad \cdot^{b}|\, Q \;\frac{P \xrightarrow{b} R}{P \mid Q \xrightarrow{b} R \mid \mathsf{push}^* Q} \qquad \cdot|^{\tau}_{y}\cdot \;\frac{P \xrightarrow{\underline{x}} R \qquad Q \xrightarrow{\overline{x}\langle y\rangle} S}{P \mid Q \xrightarrow{\tau} (\mathsf{pop}\ y)^* R \mid S}$$

$$\overline{\nu}\cdot \;\frac{P \xrightarrow{\overline{(x+1)}\langle 0\rangle} R}{\nu P \xrightarrow{\overline{x}} R} \qquad \cdot|^{\tau}_{\nu}\cdot \;\frac{P \xrightarrow{\underline{x}} R \qquad Q \xrightarrow{\overline{x}} S}{P \mid Q \xrightarrow{\tau} \nu(R \mid S)} \qquad \nu^{c}\cdot \;\frac{P \xrightarrow{\mathsf{push}^* c} R}{\nu P \xrightarrow{c} \nu R}$$

$$\nu^{b}\cdot \;\frac{P \xrightarrow{\mathsf{push}^* b} R}{\nu P \xrightarrow{b} \nu(\mathsf{swap}^* R)} \qquad !\cdot \;\frac{P \mid !P \xrightarrow{a} R}{!P \xrightarrow{a} R}$$

Figure 1: Labelled transition rules ($P+\cdot$, $P\,|^{b}\,\cdot$, $P\,|^{c}\,\cdot$, $\cdot\,|^{\tau}_{\nu}\,\cdot$ and $\cdot\,|^{\tau}_{y}\,\cdot$ variants omitted)

1. The use of push in the $\cdot\,|^{b}\,Q$ rule corresponds to the usual side-condition asserting that the binder being propagated by P is not free in Q. In the de Bruijn setting every binder "locally" has the name 0, and so this requirement can be operationalised by rewiring Q so that the name 0 is reserved. The push will be matched by a later pop which substitutes for 0, in the event that the action has a successful rendezvous.
2. The $\overline{\nu}\cdot$ rule requires an extrusion to be initiated by an output of the form $\overline{x+1}\langle 0\rangle$, capturing the usual side-condition that the name being extruded *on* is distinct from the name being extruded.
3. The rules of the form ν^{a} require that the action being propagated has the form $\mathsf{push}^* a$, ensuring that it contains no uses of index 0. This corresponds to the usual requirement that an action can only propagate through a binder that it does not mention.

The use of swap in the ν^{b} case follows Hirschkoff [14] and has no counterpart outside of the de Bruijn setting. As a propagating binder passes through another binder, their local names are 0 and 1. Propagation transposes the binders, and so to preserve naming we rewire R with a "braid" that swaps 0 and 1. Since binders are also reordered by *permutations* that relate causally equivalent executions, the swap renaming will also play an important role when we consider concurrent transitions (§3).

The following schematic derivation shows how the compact notation works. Suppose $E : P \xrightarrow{\overline{z+2}\langle 0\rangle} R$ takes place immediately under a ν-binder, causing the scope of the binder to be extruded. Then suppose the resulting bound output propagates through another binder, giving the partial derivation on the left:

$$\nu^{\overline{z}}\cdot \;\frac{\overline{\nu}\cdot \;\dfrac{E \;\dfrac{\vdots}{P \xrightarrow{\overline{z+2}\langle 0\rangle} R}}{\nu P \xrightarrow{\overline{z+1}} R}}{\nu\nu P \xrightarrow{\overline{z}} \nu R} \qquad \nu^{\overline{z}}\cdot \;\frac{\overline{\nu}E \;\dfrac{\vdots}{\nu P \xrightarrow{\overline{z+1}} R}}{\nu\nu P \xrightarrow{\overline{z}} \nu R} \qquad \nu^{\overline{z}}\overline{\nu}E \;\frac{\vdots}{\nu\nu P \xrightarrow{\overline{z}} \nu R}$$

with E standing in for the rest of the derivation. The blue constructors annotating the left-hand side of the derivation tree can be thought of as a partially unrolled "transition term" representing the proof. The

· placeholders associated with each constructor are conceptually filled by the transition terms annotating the premises of that step. We can "roll up" the derivation by a single step, by moving the premises into their corresponding placeholders, as shown in the middle figure.

By repeating this process, we can write the whole derivation compactly as $\nu^{\overline{z}}\overline{v}E$, as shown on the right. Thus the compact form is simply a flattened transition derivation: similar to a simply-typed lambda calculus term written as a conventional expression, in a (Church-style) setting where a term is, strictly speaking, a typing derivation.

2.2.1 Residuals of transitions and renamings

A transition survives any suitably-typed renaming. As alluded to already, this will be essential to formalising causal equivalence. First we define the (rather trivial) residual of a renaming $\rho : \Gamma \longrightarrow \Delta$ after an action $a : \mathsf{Action}\,\Gamma$.

Definition 1 (Residual of ρ after a)**.**

$$\begin{aligned} \rho/b &\stackrel{\text{def}}{=} \rho + 1 \\ \rho/c &\stackrel{\text{def}}{=} \rho \end{aligned}$$

The complementary residual a/ρ is also defined and is simply the renamed action $\rho^* a$ defined earlier in §2.1.1. We use the latter notation.

Lemma 9. *Suppose $E : P \xrightarrow{\quad a \quad} Q$ and $\rho : \Gamma \longrightarrow \Delta$, where $\Gamma \vdash P$. Then there exists a transition $E/\rho : \rho^* P \xrightarrow{\quad \rho^* a \quad} (\rho/a)^* Q$ such that $\mathit{tgt}(E/\rho) = \rho/a^* Q$.*

$$\begin{array}{ccc} P & \xrightarrow{\quad E \quad} & Q \\ {\scriptstyle \rho^*}\downarrow & & \downarrow{\scriptstyle (\rho/a)^*} \\ \rho^* P & \xrightarrow[E/\rho]{} & (\rho/a)^* Q \end{array}$$

The proof is the obvious lifting of a renaming to a transition, and is given in Appendix C.

We would not expect E/ρ to be derivable for arbitrary ρ in all extensions of the π-calculus. In particular, the mismatch operator $[x \neq y]P$ that steps to P if x and y are distinct names is only stable under injective renamings.

2.2.2 Structural congruences

We believe our semantics to be closed under the usual π-calculus congruences, but have not attempted to formalise this. The "braiding" congruence $\cong$ introduced in §3.2.1 is in fact a standard π-calculus congruence, which we use to track changes in the relative position of binders under permutations of traces. This could be generalised to include more congruences, but at a corresponding cost in formalisation complexity.

3 Concurrency and residuals

We now use the compact notation for derivations to define a notion of *concurrency* for transitions with the same source state, following the work of Boudol and Castellani for CCS [4]. Concurrent transitions are independent, or causally unordered: they can execute in either order without significant interference. Permutation of concurrent transitions induces a congruence on traces, which is the topic of §4.

3.1 Concurrent transitions

Transitions $P \xrightarrow{a} R$ and $Q \xrightarrow{a'} S$ are *coinitial* iff $P = Q$. We now define a symmetric and irreflexive relation $\smile$ over coinitial transitions. If $E \smile E'$ we say E and E' are *concurrent*. The relation is defined as the symmetric closure of the rules given in Figure 2, again with trivial variants of the rules omitted. For the transition constructors of the form $\cdot\,{}^{a}|\,Q$ and $\nu^{a}\cdot$ which come in bound and non-bound variants, we abuse notation a little and write a single $\smile$ rule quantified over a to mean that there are two separate (but otherwise identical) cases.

$E \smile E'$

$$\frac{}{P\,|^{a}\,F \smile E\,{}^{a'}|\,Q} \qquad \frac{E \smile E'}{E\,{}^{a}|\,Q \smile E'\,|^{\tau}_{y}\,F} \qquad \frac{F \smile F'}{P\,|^{a}\,F \smile E\,|^{\tau}_{y}\,F'} \qquad \frac{E \smile E'}{E\,{}^{a}|\,Q \smile E'\,|^{\tau}_{\nu}\,F} \qquad \frac{F \smile F'}{P\,|^{a}\,F \smile E\,|^{\tau}_{\nu}\,F'}$$

$$\frac{E \smile E'}{E + Q \smile E' + Q} \qquad \frac{F \smile F'}{P\,|^{a}\,E \smile P\,|^{a'}\,E'} \qquad \frac{E \smile E'}{E\,{}^{a}|\,Q \smile E'\,{}^{a'}|\,Q} \qquad \frac{E \smile E' \qquad F \smile F'}{E\,|^{\tau}_{y}\,F \smile E'\,|^{\tau}_{z}\,F'}$$

$$\frac{E \smile E' \qquad F \smile F'}{E\,|^{\tau}_{y}\,F \smile E'\,{}^{\tau}_{z}|\,F'} \qquad \frac{E \smile E' \qquad F \smile F'}{E\,|^{\tau}_{y}\,F \smile E'\,|^{\tau}_{\nu}\,F'} \qquad \frac{E \smile E' \qquad F \smile F'}{E\,|^{\tau}_{y}\,F \smile E'\,{}^{\tau}_{\nu}|\,F'} \qquad \frac{E \smile E' \qquad F \smile F'}{E\,|^{\tau}_{\nu}\,F \smile E'\,|^{\tau}_{\nu}\,F'}$$

$$\frac{E \smile E' \qquad F \smile F'}{E\,|^{\tau}_{\nu}\,F \smile E'\,{}^{\tau}_{\nu}|\,F'} \qquad \frac{E \smile E'}{\overline{\nu}E \smile \overline{\nu}E'} \qquad \frac{E \smile E'}{\overline{\nu}E \smile \nu^{a}E'} \qquad \frac{E \smile E'}{\nu^{a}E \smile \nu^{a'}E'} \qquad \frac{E \smile E'}{!E \smile !E'}$$

Figure 2: Concurrent coinitial transitions ($P + \cdot$, and some $\cdot\,{}^{\tau}_{y}|\cdot$ and $\cdot\,{}^{\tau}_{\nu}|\cdot$ variants omitted)

The first rule, $P\,|^{a}\,F \smile E\,{}^{a'}|\,Q$, says that two transitions E and F are concurrent if they take place on opposite sides of the same parallel composition. The remaining rules propagate concurrent sub-transitions up through ν, choice, parallel composition, and replication. Note that there are no rules allowing us to conclude that a left-choice step is concurrent with a right-choice step: choices are mutually exclusive. Likewise, there are no rules allowing us to conclude that an input or output transition is concurrent with any other transition; since both E and E' are required to be coinitial, if one of them is an input or output step then they are equal and hence not concurrent.

The $E\,|^{\tau}_{y}\,F \smile E'\,|^{\tau}_{z}\,F'$ rule says that a rendezvous is concurrent with another rendezvous under the same parallel composition, as long as the two inputs are concurrent on the left, and the two outputs are concurrent on the right. The $E\,|^{\tau}_{y}\,F \smile E'\,{}^{\tau}_{z}|\,F'$ variant is similar, but permits concurrent input and output on the left, with their rendezvous partners concurrent on the right. The $E\,|^{\tau}_{y}\,F \smile E'\,{}^{\tau}_{\nu}|\,F'$ rule and variants permit a regular rendezvous and an extrusion-rendezvous to be concurrent.

3.2 Residuals of concurrent transitions

Intuitively, if $E \smile E'$ then E and E' are "parallel moves" in the sense of Curry and Feys [9]: if either execution step is taken, the other remains valid, and if both are taken, one ends up in (essentially) the same state, regardless of which step is taken first.

However, concurrent transitions are not completely independent: the location and nature of the redex identified by one transition may change as a consequence of the earlier transition. This intuition is captured by the notion of the residual E/E', explored notably by Lévy in the lambda calculus [16],

and later considered by Stark for concurrent transition systems [25] and in the specific setting of CCS by Boudol and Castellani [4]. The residual specifies how E must be adjusted to take into account the fact that E' has taken place.

Definition 2 (Residual)**.** Suppose $E \smile E'$. Then the *residual* of E after E', written E/E', is given by the least function satisfying the equations in Figure 3.

The operator $\cdot/\cdot$ has higher precedence than any transition constructor. The definition makes use of the renaming lemmas in §2.1.2, and is rather tricky; Appendix C.1 gives several examples which illustrate some of the subtleties that arise in the π-calculus setting, in particular relating to name extrusion.

E/E'

$$
\begin{aligned}
(P \mid^a F)/(E\,{}^c\!\mid Q) &= \mathsf{tgt}(E) \mid^a F \\
(P \mid^a F)/(E\,{}^b\!\mid Q) &= \mathsf{tgt}(E) \mid^a \mathsf{push}^* F \\
(E\,{}^a\!\mid Q)/(P \mid^c F) &= E\,{}^a\!\mid \mathsf{tgt}(F) \\
(E\,{}^a\!\mid Q)/(P \mid^b F) &= \mathsf{push}^* E\,{}^a\!\mid \mathsf{tgt}(F) \\
(E\,{}^a\!\mid Q)/(E' \mid^\tau_y F) &= (\mathsf{pop}\ y)^*(E/E')\,{}^a\!\mid \mathsf{tgt}(F) \\
(P \mid^a F)/(E \mid^\tau_y F') &= (\mathsf{pop}\ y)^*\mathsf{tgt}(E) \mid^a F/F' \\
(E \mid^\tau_y F)/(E'\,{}^b\!\mid Q) &= E/E' \mid^\tau_y \mathsf{push}^* F \\
(E \mid^\tau_y F)/(E'\,{}^c\!\mid Q) &= E/E' \mid^\tau_y F \\
(E \mid^\tau_y F)/(P \mid^b F') &= \mathsf{push}^* E \mid^\tau_y F/F' \\
(E \mid^\tau_y F)/(P \mid^c F') &= E \mid^\tau_y F/F' \\
(E\,{}^{\underline{x}}\!\mid Q)/(E' \mid^\tau_\nu F) &= \nu^{\underline{x}}(E/E'\,{}^{\underline{x+1}}\!\mid \mathsf{tgt}(F)) \\
(E\,{}^{\overline{x}}\!\mid Q)/(E' \mid^\tau_\nu F) &= \overline{\nu}(E/E'\,{}^{\overline{x+1}\langle 0\rangle}\!\mid \mathsf{tgt}(F)) \\
(E\,{}^c\!\mid Q)/(E' \mid^\tau_\nu F) &= \nu^c(E/E'\,{}^{\mathsf{push}^* c}\!\mid \mathsf{tgt}(F)) \\
(P \mid^{\underline{x}} F)/(E \mid^\tau_\nu F') &= \nu^{\underline{x}}(\mathsf{tgt}(E) \mid^{\underline{x+1}} F/F') \\
(P \mid^{\overline{x}} F)/(E \mid^\tau_\nu F') &= \overline{\nu}(\mathsf{tgt}(E) \mid^{\overline{x+1}\langle 0\rangle} F/F') \\
(P \mid^c F)/(E \mid^\tau_\nu F') &= \nu^c(\mathsf{tgt}(E) \mid^{\mathsf{push}^* c} F/F') \\
(E \mid^\tau_\nu F)/(E'\,{}^b\!\mid Q) &= E/E' \mid^\tau_\nu \mathsf{push}^* F \\
(E \mid^\tau_\nu F)/(E'\,{}^c\!\mid Q) &= E/E' \mid^\tau_\nu F \\
(E \mid^\tau_\nu F)/(P \mid^{\underline{x}} F') &= \mathsf{push}^* E \mid^\tau_\nu F/F' \\
(E \mid^\tau_\nu F)/(P \mid^{\overline{x}} F') &= \mathsf{push}^* E \mid^\tau_0 F/F' \\
(E \mid^\tau_\nu F)/(P \mid^c F') &= E \mid^\tau_\nu F/F' \\
(E + Q)/(E' + Q) &= E/E' \\
(P \mid^{\underline{x}} F)/(P \mid^b F') &= \mathsf{push}^* P \mid^{\underline{x}} F/F' \\
(P \mid^b F)/(P \mid^{\underline{x}} F') &= \mathsf{push}^* P \mid^b F/F' \\
(P \mid^{\overline{x}} F)/(P \mid^{\overline{u}} F') &= \mathsf{push}^* P \mid^{\overline{x+1}\langle 0\rangle} F/F' \\
(P \mid^c F)/(P \mid^b F') &= \mathsf{push}^* P \mid^c F/F' \\
(P \mid^a F)/(P \mid^c F') &= P \mid^a F/F' \\
(E\,{}^{\underline{x}}\!\mid Q)/(E'\,{}^b\!\mid Q) &= E/E'\,{}^{\underline{x}}\!\mid \mathsf{push}^* Q \\
(E\,{}^b\!\mid Q)/(E'\,{}^{\underline{x}}\!\mid Q) &= E/E'\,{}^b\!\mid \mathsf{push}^* Q \\
(E\,{}^{\overline{x}}\!\mid Q)/(E'\,{}^{\overline{u}}\!\mid Q) &= E/E'\,{}^{\overline{x+1}\langle 0\rangle}\!\mid \mathsf{push}^* Q \\
(E\,{}^c\!\mid Q)/(E'\,{}^b\!\mid Q) &= E/E'\,{}^c\!\mid \mathsf{push}^* Q \\
(E\,{}^a\!\mid Q)/(E'\,{}^c\!\mid Q) &= E/E'\,{}^a\!\mid Q \\
(E \mid^\tau_y F)/(E' \mid^\tau_z F') &= (\mathsf{pop}\ z)^*(E/E') \mid^\tau_y F/F' \\
(E \mid^\tau_y F)/(E' \mid^\tau_\nu F') &= \nu^\tau(E/E' \mid^\tau_y F/F') \\
(E \mid^\tau_\nu F)/(E' \mid^\tau_z F') &= (\mathsf{pop}\ z)^*(E/E') \mid^\tau_\nu F/F' \\
(E \mid^\tau_\nu F)/(E' \mid^\tau_\nu F') &= \nu^\tau(E/E' \mid^\tau_\nu F/F') \\
(\overline{\nu}E)/(\overline{\nu}E') &= E/E' \\
(\overline{\nu}E)/(\nu^b E') &= \overline{\nu}\ \mathsf{swap}^*(E/E') \\
(\overline{\nu}E)/(\nu^c E') &= \overline{\nu}\ E/E' \\
(\nu^b E)/(\overline{\nu}E') &= E/E' \\
(\nu^c E)/(\overline{\nu}E') &= E/E' \\
(\nu^b E)/(\nu^b E') &= \nu E/E' \\
(\nu^c E)/(\nu^b E') &= \nu^c\ \mathsf{swap}^*(E/E') \\
(\nu^b E)/(\nu^c E') &= \nu^b\ E/E' \\
(\nu^c E)/(\nu^c E') &= \nu^c\ E/E' \\
(!E)/(!E') &= E/E'
\end{aligned}
$$

Figure 3: Residual of E after E', omitting $\cdot \mid^\tau_y \cdot$ and $\cdot \mid^\tau_\nu \cdot$ cases

3.2.1 Cofinality of residuals

The idea that one ends up in the same state regardless of whether E or E' is taken first is called *cofinality*. In CCS, where actions never involve binders, and in the lambda calculus, where binders do not move around, cofinality simply means the target states are equivalent. Things are not quite so simple in late-style π-calculus, because binders propagate during execution, as bound actions. Consider the process $\underline{x}.P \mid \underline{z}.Q$ with two concurrent input actions. Initiating one of the inputs (say $\underline{x}$) starts propagating a

binder. As this binder passes through the parallel composition, the transition rules use push to "reserve" the free variable 0 in the right half of the process for potential use by a subsequent pop:

$$\cdot \overset{\underline{x}}{|}\, \underline{z}.Q \quad \frac{\Gamma \vdash \underline{x}.P \xrightarrow{\underline{x}} \Gamma + 1 \vdash P}{\Gamma \vdash \underline{x}.P \mid \underline{z}.Q \xrightarrow{\underline{x}} \Gamma + 1 \vdash P \mid \underline{z+1}.(\mathsf{push}+1)^* Q}$$

When the action $(\underline{z+1})$ is performed, a push on the left leaves the final state with both 0 and 1 reserved:

$$P \overset{\underline{z+1}}{|}\, \cdot \quad \frac{\Gamma + 1 \vdash \underline{z+1}.(\mathsf{push}+1)^* Q \xrightarrow{\underline{z+1}} \Gamma + 2 \vdash (\mathsf{push}+1)^* Q}{\Gamma + 1 \vdash P \mid \underline{z+1}.(\mathsf{push}+1)^* Q \xrightarrow{\underline{z+1}} \Gamma + 2 \vdash \mathsf{push}^* P \mid (\mathsf{push}+1)^* Q}$$

Had these concurrent actions happened in the opposite order, the push on the left would have been applied first. The final state would be $(\mathsf{push}+1)^* P \mid \mathsf{push}^* Q$, which is the image of $\mathsf{push}^* P \mid (\mathsf{push}+1)^* Q$ in the permutation swap which renames 0 to 1 and 1 to 0. Instead of the usual cofinality square, the final states are related by a "braid" (in the form of a swap) which permutes the free names:

$$\begin{array}{ccc}
\Gamma + 1 \vdash P \mid \underline{z+1}.(\mathsf{push}+1)^* Q & \xrightarrow{\underline{z+1}} & \Gamma + 2 \vdash \mathsf{push}^* P \mid (\mathsf{push}+1)^* Q \\
\overset{\underline{x}}{\nearrow} & & \downarrow \mathsf{swap}^* \\
\Gamma \vdash \underline{x}.P \mid \underline{z}.Q & & \Gamma + 2 \vdash \mathsf{swap}^* \mathsf{push}^* P \mid \mathsf{swap}^* (\mathsf{push}+1)^* Q \\
\underset{\underline{z}}{\searrow} & & \| \alpha \mid \beta \\
\Gamma + 1 \vdash \underline{x+1}.(\mathsf{push}+1)^* P \mid Q & \xrightarrow[\underline{x+1}]{} & \Gamma + 2 \vdash (\mathsf{push}+1)^* P \mid \mathsf{push}^* Q
\end{array}$$

Here α and β are equalities obtained from Lemma 5.

It is not just the reordering of bound actions which nuances π-calculus cofinality. When two τ actions are reordered, which happen to be extrusion rendezvous of distinct binders, the resulting binders exchange positions in the final process. In the standard π-calculus this would be subsumed by the congruence $(\nu xy)\, P \cong (\nu yx)\, P$. In the de Bruijn setting, where adjacent binders cannot be distinguished, the analogous rule is $\nu\nu P \cong \nu\nu(\mathsf{swap}^* P)$, which applies a swap braid under the two binders.

These two possibilities are subsumed by the following generalised notion of cofinality. First we define a braiding congruence $\cong$ just large enough to permit swap under a pair of binders. "Cofinality" is then defined using a more general braiding relation which additionally permits swaps of free variables. Examples showing reordered extrusions are given in Appendix C.1, including concurrent extrusions of the *same* binder, an interesting case identified by Cristescu et al. [8].

Definition 3 (Braiding congruence)**.** Inductively define the binary relation $\cong$ over processes using the rules given in Figure 4.

In Figure 4, rule names are shown to the left in blue, permitting a compact term-like notation for $\cong$ proofs similar to the convention we introduced earlier for transitions. The process constructors are overloaded to witness compatibility; transitivity is denoted by $\circ$. It is easy to see that $\cong$ is also reflexive and symmetric, and therefore a congruence. $P_{\cong}$ denotes the canonical proof that $P \cong P$.

In what follows ϕ and ψ range over braiding congruences; $\mathrm{src}(\phi)$ and $\mathrm{tgt}(\phi)$ denote P and R, for any $\phi : P \cong R$. As with transitions, braiding congruences are stable under renamings, giving rise to the usual notion of residuation; however ρ/ϕ is always ρ. The proof is a straightforward induction.

$P \cong R$

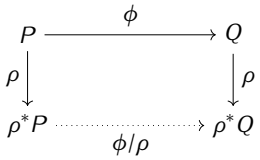
$$\nu\nu\text{-swap}_P \ \frac{}{\nu\nu(\text{swap}^* P) \cong \nu\nu P} \qquad \nu\nu\text{-swap}_P^{-1} \ \frac{}{\nu\nu P \cong \nu\nu(\text{swap}^* P)} \qquad \cdot \circ \cdot \ \frac{R \cong S \qquad P \cong R}{P \cong S} \qquad \mathbf{0} \ \frac{}{\mathbf{0} \cong \mathbf{0}}$$

$$\underline{x}.\cdot \ \frac{P \cong R}{\underline{x}.P \cong \underline{x}.R} \qquad \overline{x}\langle y\rangle.\cdot \ \frac{P \cong R}{\overline{x}\langle y\rangle.P \cong \overline{x}\langle y\rangle.R} \qquad \cdot + \cdot \ \frac{P \cong R \qquad Q \cong S}{P + Q \cong R + S} \qquad \cdot \mid \cdot \ \frac{P \cong R \qquad Q \cong S}{P \mid Q \cong R \mid S}$$

$$\nu\cdot \ \frac{P \cong R}{\nu P \cong \nu R} \qquad !\cdot \ \frac{P \cong R}{!P \cong !R}$$

Figure 4: Braiding congruence $\cong$

Lemma 10. *For any $\Gamma \vdash P$, suppose $\phi : P \longrightarrow Q$ and $\rho : \Gamma \longrightarrow \Delta$. Then there exists a braiding congruence $\phi/\rho : \rho^* P \longrightarrow \rho^* Q$.*



Definition 4 (Braiding)**.** For any $\Delta \in \{0, 1, 2\}$ define the following family of bijective renamings $\text{braid}_{\Gamma,\Delta} : \Gamma + \Delta \longrightarrow \Gamma + \Delta$ and symmetric *braiding* relations $\bowtie_{\Gamma,\Delta}$ over processes in $\Gamma + \Delta$.

$$\begin{array}{l} \text{braid}_{\Gamma,0} = \text{id}_\Gamma : \Gamma \longrightarrow \Gamma \\ \text{braid}_{\Gamma,1} = \text{id}_{\Gamma+1} : \Gamma + 1 \longrightarrow \Gamma + 1 \\ \text{braid}_{\Gamma,2} = \text{swap}_\Gamma : \Gamma + 2 \longrightarrow \Gamma + 2 \end{array} \qquad P \bowtie_{\Gamma,\Delta} P' \iff \text{braid}_{\Gamma,\Delta}{}^* P \cong P'$$

Our key soundness result is that residuals of concurrent transitions E and E' are always cofinal up to a braiding of type $\bowtie_{\Gamma,\Delta}$ where $\Delta \in \{0, 1, 2\}$ is the number of free variables introduced by E and E'/E. Rather than the usual parallel-moves square on the left, the residuals satisfy pentagons of the form shown in the centre of Figure 5, where $\gamma : Q \bowtie_{\Gamma,\Delta} Q'$ is a braiding.

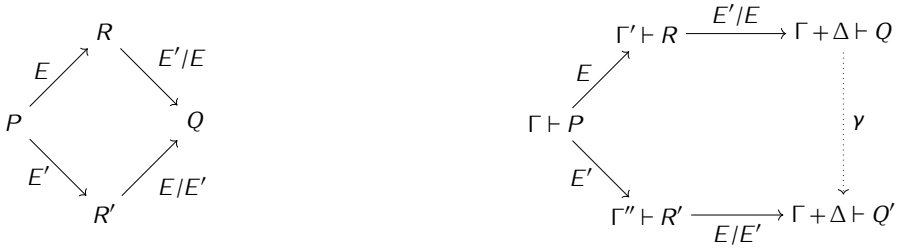


Figure 5: Cofinality in the style of CCS (left); with explicit braiding (right)

Arranging for this to hold by construction introduces a certain amount of complexity, so we prove cofinality as a separate theorem.

Theorem 1 (Cofinality of residuals)**.** *Suppose E and E' are the transitions on the right of Figure 5, with $E \smile E'$. Then there exists $cofin_{E,E'} : Q \bowtie_\Delta Q'$.*

The notion of concurrency extends into dimensions greater than two. Following Pratt's higher-dimensional automata [23], we can consider a proof $\chi : E \smile E'$ as a surface that represents the concurrency of E and E' without committing to an order of occurrence. Every such $\chi : E \smile E'$ has a two-dimensional residual χ/E'' with respect to a third concurrent transition E''. First we note that concurrent transitions are closed under renamings.

Lemma 11. *Suppose $\rho : \Gamma \longrightarrow \Delta$ and E, E' are both transitions from $\Gamma \vdash P$, with $\chi : E \smile E'$. Then there exists $\chi/\rho : E/\rho \smile E'/\rho$.*

Proof. By induction on χ, using Lemma 9. □

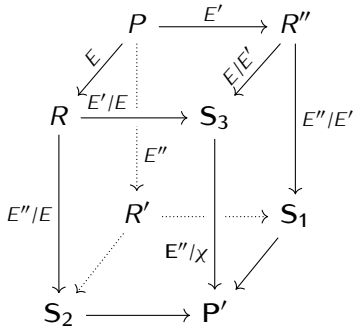
Theorem 2 (Residuation preserves concurrency)**.**
Suppose $\chi : E \smile E'$ with $E \smile E''$ and $E \smile E''$. Then there exists $\chi/E'' : E/E'' \smile E'/E''$.

Proof. By induction on χ and inversion on the other two derivations, using Lemma 11. □

Theorem 3. *Suppose $\chi : E \smile E'$, with $E' \smile E''$ and $E'' \smile E$. Then:*

$$((E'/E'')/(E/E''))/cofin_{E'',E} = (E'/E)/(E''/E)$$

The diagram below illustrates Theorems 2 and 3 informally. The three faces χ, χ' and χ'' with P as a vertex witness the pairwise concurrency of E, E' and E''. Theorem 2 ensures that these have opposite faces χ/E'', χ'/E and χ''/E'. Theorem 3 states that, up to a suitable braiding, there is a unique residual of a one-dimensional transition after two-dimensional one, connecting the faces χ'/E and χ''/E' via the shared edge E''/χ. Analogous reasoning for E/χ' and E'/χ'' yields a cubical transition with target $\mathbf{P}'$.



The bold font for $\mathbf{S_1}$, $\mathbf{S_1}$ and $\mathbf{S_1}$ indicates that they represent not a unique process but a permutation group of processes related by braidings. At $\mathbf{P}'$ there are potentially $3! = 6$ variants of the target process, one for each possible interleaving of E, E' and E''. The notation $\mathbf{E}''/\chi$ is again informal, referring not to a unique transition but to a permutation group related by braidings.

4 Causal equivalence

4.1 Traces

Define Action* Γ to be the set of finite sequences of composable actions starting at Γ. The empty sequence at Γ is written $[]$; extension to the left is written $a :: \widetilde{a}$. A *trace* $t : P \xrightarrow{\widetilde{a}} R$ is a finite sequence of composable transitions with initial state $\mathrm{src}(t) = P$ and final state $\mathrm{tgt}(t) = R$. The empty trace at P is written $[]_P$; extension to the left of $t : R \xrightarrow{\widetilde{a}} S$ by $E : P \xrightarrow{a} R$ is written $E :: t$.

4.2 Residuals of traces and braidings

To define the residual of a trace t with respect to a braiding γ, we first observe that a braiding congruence $\phi : P \cong P'$ commutes (on the nose) with a transition $E : P \xrightarrow{a} Q$, inducing the corresponding notions of residual ϕ/E (the image of the braiding congruence in the transition) and E/ϕ (the image of the transition in the braiding congruence).

Theorem 4. *Suppose $E : P \xrightarrow{a} R$ and $\phi : P \cong P'$. Then there exists a process R', transition $E/\phi : P' \xrightarrow{a} R'$ and structural congruence $\phi/E : R \cong R'$.*

$$\begin{array}{ccc} P & \xrightarrow{\quad E \quad} & R \\ {\scriptstyle\phi}\downarrow & & \vdots\,{\scriptstyle\phi/E} \\ P' & \xrightarrow[E/\phi]{} & R' \end{array}$$

Proof. By the defining equations in Figure 6.

Unlike residuals of the form E/E', the cofinality of E/ϕ and ϕ/E is by construction. Appendix C.2 illustrates cofinality for the cases where ϕ is of the form $\nu\nu$-swap$_P$.

To extend this notion of residuation from braiding congruences to braidings requires a more general notion of braiding which permits the renaming component of the braiding to be shifted under a binder. First recall (from Definition 4) that any braiding $\gamma : P \bowtie_{\Gamma,\Delta} P'$ is of the form $\phi \circ \mathrm{braid}_{\Gamma,\Delta}$, where $\mathrm{braid}_{\Gamma,\Delta} : \Gamma + \Delta \longrightarrow \Gamma + \Delta$ is the renaming id or swap, as determined by $\Delta \in \{0,1,2\}$, and ϕ is a braiding congruence. We omit the Γ, Δ subscripts whenever possible. The more general form of braiding allows the braid and ϕ components to be translated by an arbitrary context Δ'.

Definition 5 (Δ-shifted braiding)**.** For any context Δ define

$$P \bowtie^{\Delta}_{\Gamma,\Delta'} P' \iff (\mathrm{braid}_{\Gamma,\Delta'} + \Delta)^* P \cong P'$$

Now we define the residual of a transition $E : \Gamma \vdash P \xrightarrow{a} \Gamma + \Delta \vdash R$, where $\Delta \in \{0,1\}$, and coinitial braiding γ and show that the residual γ/E is γ shifted by Δ.

Definition 6 (Residuals of transitions and braidings)**.** For any transition $E : P \xrightarrow{a} R$ and braiding $\gamma : P \bowtie^{\Delta} P'$ with $\gamma = \phi \circ \sigma$, define E/γ and γ/E by the following equations.

$$E/(\phi \circ \sigma) = (E/\sigma)/\phi \qquad\qquad (\phi \circ \sigma)/E = (\phi/(E/\sigma)) \circ \sigma/a$$

Cofinality is immediate by composing the square obtained by applying Lemma 9 to E and σ with the square obtained from Theorem 4 above to ϕ and E/σ. Closure of (Δ-shifted) braidings under residuation follows from the fact that $\sigma/a = \sigma + \Delta'$ for some $\Delta' \in \{0,1\}$.

E/ϕ

$$
\begin{aligned}
\nu\nu\text{-swap}_{\mathrm{src}(E)}/(\overline{\nu}\nu^{\overline{x+1}\langle 0\rangle}E) &= \nu^{\overline{x}}\overline{\nu}(\mathrm{swap}^*E)\\
\nu\nu\text{-swap}_{\mathrm{src}(E)}/(\nu^{\overline{x}}\overline{\nu}E) &= \overline{\nu}\nu^{\overline{x+1}\langle 0\rangle}(\mathrm{swap}^*E)\\
\nu\nu\text{-swap}_{\mathrm{src}(E)}/(\nu^c\nu^{c'}E) &= \nu^c\nu^{c'}(\mathrm{swap}^*E)\\
\nu\nu\text{-swap}_{\mathrm{src}(E)}/(\nu^b\nu^{b'}E) &= \nu^b\nu^{b'}(\mathrm{swap}^*E)\\
(\underline{x}.P)/(\underline{x}.\phi) &= \underline{x}.\mathrm{tgt}(\phi)\\
(\overline{x}\langle y\rangle.P)/(\overline{x}\langle y\rangle.\phi) &= \overline{x}\langle y\rangle.\mathrm{tgt}(\phi)\\
(E+Q)/(\phi+\psi) &= E/\phi+\mathrm{tgt}(\psi)\\
(E\,{}^b|\,Q)/(\phi+\psi) &= E/\phi\,{}^b|\,\mathrm{tgt}(\psi)\\
(E\,{}^c|\,Q)/(\phi+\psi) &= E/\phi\,{}^c|\,\mathrm{tgt}(\psi)\\
(P\,|^b\,F)/(\phi+\psi) &= \mathrm{tgt}(\phi)\,|^b\,F/\psi\\
(P\,|^c\,F)/(\phi+\psi) &= \mathrm{tgt}(\phi)\,|^c\,F/\psi\\
(E\,|^\tau_y\,F)/(\phi\mid\psi) &= E/\phi\,|^\tau_y\,F/\psi\\
(E\,|^\tau_\nu\,F)/(\phi\mid\psi) &= E/\phi\,|^\tau_\nu\,F/\psi\\
(\overline{\nu}E)/(\nu\phi) &= \overline{\nu}\;E/\phi\\
(\nu^bE)/(\nu\phi) &= \nu^bE/\phi\\
(\nu^cE)/(\nu\phi) &= \nu^cE/\phi\\
(!E)/(!\phi) &= !E/(\phi\mid !\phi)\\
E/(\phi'\circ\phi) &= (E/\phi)/\phi'
\end{aligned}
$$

ϕ/E

$$
\begin{aligned}
\nu\nu\text{-swap}_{\mathrm{src}(E)}/(\overline{\nu}\nu^{\overline{x+1}\langle 0\rangle}E) &= \nu\;\mathrm{tgt}(E)_{\cong}\\
\nu\nu\text{-swap}_{\mathrm{src}(E)}/(\nu^{\overline{x}}\overline{\nu}E) &= \nu\;\mathrm{swap}^*\mathrm{tgt}(E)_{\cong}\\
\nu\nu\text{-swap}_{\mathrm{src}(E)}/(\nu^c\nu^{c'}E) &= \nu\nu\text{-swap}^{-1}_{\mathrm{tgt}(E)}\\
\nu\nu\text{-swap}_{\mathrm{src}(E)}/(\nu^b\nu^{b'}E) &= \nu\nu\text{-swap}^{-1}_{\mathrm{swap}^*(\mathrm{swap}+1)^*\mathrm{swap}^*\mathrm{tgt}(E)}\\
(\underline{x}.\phi)/(\underline{x}.P) &= \phi\\
(\overline{x}\langle y\rangle.\phi)/(\overline{x}\langle y\rangle.P) &= \phi\\
(\phi+\psi)/(E+Q) &= \phi/E\\
(\phi+\psi)/(E\,{}^b|\,Q) &= \phi/E\mid\mathrm{push}^*\psi\\
(\phi+\psi)/(E\,{}^c|\,Q) &= \phi/E\mid\psi\\
(\phi+\psi)/(P\,|^b\,F) &= \mathrm{push}^*\phi\mid\psi/F\\
(\phi+\psi)/(P\,|^c\,F) &= \phi\mid\psi/F\\
(\phi\mid\psi)/(E\,|^\tau_y\,F) &= (\mathrm{pop}\;y)^*\phi/E\\
(\phi\mid\psi)/(E\,|^\tau_\nu\,F) &= \nu(\phi/E\mid\psi/F)\\
(\nu\phi)/(\overline{\nu}E) &= \phi/E\\
(\nu\phi)/(\nu^bE) &= \nu\;\mathrm{swap}^*\phi/E\\
(\nu\phi)/(\nu^cE) &= \nu\;\phi/E\\
(!\phi)/(!E) &= (\phi\mid !\phi)/E\\
(\phi'\circ\phi)/E &= (\phi'/(E/\phi))\circ\phi/E
\end{aligned}
$$

Figure 6: Residual of transition E and coinitial braiding congruence ϕ

$$
\begin{array}{ccc}
P & \xrightarrow{\quad E\quad} & R\\
{\scriptstyle\sigma+\Delta}\downarrow & & \downarrow{\scriptstyle\sigma+\Delta'}\\
(\sigma+\Delta)^*P & \xrightarrow{E/(\sigma+\Delta)} & (\sigma+\Delta')^*R\\
{\scriptstyle\phi}\downarrow & & \downarrow{\scriptstyle\phi/(E/(\sigma+\Delta))}\\
P' & \xrightarrow[(E/(\sigma+\Delta))/\phi]{} & R'
\end{array}
$$

where both $E/(\sigma+\Delta)$ and $(E/(\sigma+\Delta))/\phi$ have the action $(\sigma+\Delta)^*a$.

Finally, we extend the definition to traces.

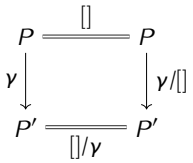
Definition 7 (Residuals of action sequences and renamings)**.**
Suppose $\rho:\Gamma\longrightarrow\Delta$ and $\widetilde{a}:\mathrm{Action}^*\,\Gamma$. Define the residuals $\rho/\widetilde{a}$ and $\widetilde{a}/\rho$, writing the latter as $\rho^*\widetilde{a}$.

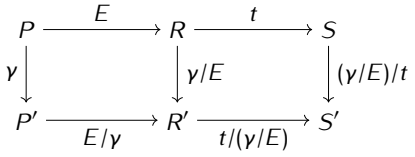
$$
\begin{aligned}
\rho/[]_\Gamma &= \rho & \qquad \rho/(a::\widetilde{a}) &= (\rho/a)/\widetilde{a}\\
\rho^*[]_\Gamma &= []_\Delta & \qquad \rho^*(a::\widetilde{a}) &= (\rho^*a)::(\rho/a)^*\widetilde{a}
\end{aligned}
$$

Lemma 12 (Residuals of traces and braidings)**.**
Suppose $t:P\xrightarrow{\widetilde{a}}R$ and $\gamma=\phi\circ\sigma:P\bowtie^\Delta P'$. Then there exists a process R', trace $P'\xrightarrow{\sigma^*\widetilde{a}}R'$ and braiding $\gamma/t:R\bowtie R'$.

$$
\begin{array}{ccc}
P & \xrightarrow{\quad t\quad} & R\\
{\scriptstyle\gamma}\downarrow & & \vdots\,{\scriptstyle\gamma/t}\\
P' & \xrightarrow[t/\gamma]{} & R'
\end{array}
$$

Proof. By the following defining equations.





$$[]_P/\gamma = []_{P'} \qquad (E :: t)/\gamma = (E/\gamma) :: t/(\gamma/E)$$
$$\gamma/[]_P = \gamma \qquad \gamma/(E :: t) = (\gamma/E)/t$$

4.3 Causal equivalence

We now define *causal equivalence*, the congruence over traces induced by the notion of transition residual from §3.2. A causal equivalence $\alpha : t \simeq u$ witnesses the reordering of one trace t into a coinitial trace u by the permutation of concurrent transitions. Meta-variables α, β range over causal equivalences.

Definition 8. Inductively define the relation $\simeq$ given by the rules in Figure 7, where syntactically $\simeq$ has lower priority than $\cdot :: \cdot$. If $\alpha : t \simeq u$ then $\mathrm{src}(\alpha)$ and $\mathrm{tgt}(\alpha)$ denote t and u respectively.

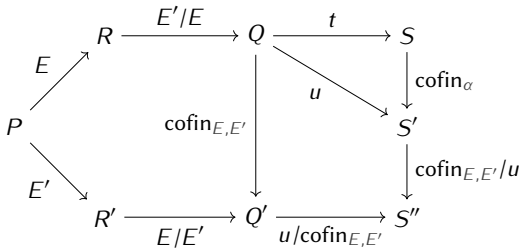
$t \simeq u$

$$[]_P \ \frac{}{[]_P \simeq []_P} \qquad \cdot :: \cdot \ \frac{E : P \xrightarrow{a} R \qquad t \simeq u}{E :: t \simeq E :: u}\ \mathrm{src}(t) = R \qquad \cdot \circ \cdot \ \frac{t' \simeq u \qquad t \simeq t'}{t \simeq u}$$

$$(\cdot :\rightleftharpoons: \cdot) :: \cdot \ \frac{E : P \xrightarrow{a} R \qquad E' : P \xrightarrow{a'} R' \qquad t \simeq u}{E :: E'/E :: t \simeq E' :: E/E' :: u/\mathrm{cofin}_{E,E'}}\ E \smile E'$$

Figure 7: Causal equivalence

The $[]_P$ and $E :: \alpha$ rules are the congruence cases. The $\alpha \circ \beta$ rule closes under transitivity, which is a form of vertical composition. The transposition rule $(E :\rightleftharpoons: E') :: \alpha$ extends an existing causal equivalence $\alpha : t \simeq u$ with the two possible interleavings of concurrent steps $E \smile E'$. What is interesting about this rule is that the trace u must be transported through the braiding $\mathrm{cofin}_{E,E'}$ witnessing the cofinality of E and E', in order to obtain a trace $u/\mathrm{cofin}_{E,E'}$ composable with E'/E. The following diagram illustrates.



As the diagram suggests, the transposition rule causes braidings to compose vertically. Here, cofin_α is a composite braiding relating S to S', which is extended by the braiding $\mathrm{cofin}_{E,E'}/u$ to relate S to S''. We leave formalising this aspect of causal equivalence to future work.

Theorem 5. $\simeq$ *is an equivalence relation.*

Proof. Reflexivity is a trivial induction, using the $[]_P$ and $E :: \alpha$ rules. Transitivity is immediate from the $\alpha \circ \beta$ rule. Symmetry is trivial in the $[]_P$, $E :: \alpha$ and $\alpha \circ \beta$ cases. The $(E :\rightleftharpoons: E') :: \alpha$ case requires the symmetry of $\smile$ and that $(u/\mathsf{cofin}_\alpha)/\mathsf{cofin}_\alpha^{-1} = u$, where $u = \mathsf{tgt}(\alpha)$.

5 Related work

Hirschkoff's μs calculus [14] has a similar treatment of de Bruijn indices. Its renaming operators $\langle x \rangle$, ϕ and ψ are effectively our pop x, push and swap renamings, but fused with the $\cdot^*$ operator which applies a renaming to a process. Hirschkoff's operators are also syntactic forms in the μs calculus, rather than meta-operations, and therefore the operational semantics also includes rules for reducing occurrences of the renaming operators that arise during a process reduction step.

Formalisations of the π-calculus have been undertaken in several theorem provers used for mechanised metatheory. Due to space limits, we limit attention to closely-related formalisation techniques based on constructive logics.

Coq. Hirschkoff [13] formalised the π-calculus in Coq using de Bruijn indices, and verified properties such as congruence and structural equivalence laws of bisimulation. Despeyroux [12] formalised the π-calculus in Coq using weak higher-order abstract syntax, assuming a decidable type of names, and using two separate transitions, for ordinary, input and output transitions respectively; for input and output transitions the right-hand side is a function of type $\mathsf{name} \longrightarrow \mathsf{proc}$. This formalisation included a simple type system and proof of type soundness. Honsell, Miculan and Scagnetto [15] formalised the π-calculus in Coq, also using weak higher-order abstract syntax. The type of names name is a type parameter assumed to admit decidable equality and freshness (notin) relations. Transitions are encoded using two inductive definitions, for free and bound actions, which differ in the type of the third argument (proc vs. $\mathsf{name} \longrightarrow \mathsf{proc}$). Numerous results from Milner, Parrow and Walker [19] are verified, using the *theory of contexts* (whose axioms are assumed in their formalisation, but have been validated semantically).

CLF. Cervesato, Pfenning, Walker and Watkins [6] formalise synchronous and asynchronous versions of π-calculus in the Concurrent Logical Framework (CLF). CLF employs higher-order abstract syntax, linearity and a monadic encapsulation of certain linear constructs that can identify objects such as traces up to causal equivalence. Thus, CLF's π-calculus encodings naturally induce equivalences on traces. However, a nontrivial effort appears necessary to compare CLF's notion of trace equivalence with others (including ours) due to the distinctive approach taken in CLF.

Agda. Orchard and Yoshida [21] present a translation from a functional language with effects to a π-calculus with session types and verify some type-preservation properties of the translation in Agda.

6 Conclusions and future work

To the best of our knowledge, we are the first to report on a formalisation of the operational behavior of the π-calculus in Agda. Compared to prior formalisations, ours is distinctive in two ways.

First, our formalisation employs an indexed family of types for process terms and uses the indices instead of binding to deal with scope extrusion. Formalisations of lambda-calculi often employ this technique, but to our knowledge only Orchard and Yoshida report a similar approach for a π-calculus formalisation. This choice helps tame the complexity of de Bruijn indices, because many invariants are automatically checked by the type system rather than requiring additional explicit reasoning.

Second, our work appears to be the first to align the notion of "proved transitions" from Boudol and Castellani's work on CCS with "transition proofs" in the π-calculus. This hinges on the capability to manipulate and perform induction or recursion over derivations, and means we can leverage dependent typing so that residuation is defined only for concurrent transitions, rather than on all pairs of transitions. It is worth noting that while CLF's approach to encoding π-calculus automatically yields an equivalence on traces, it is unclear (at least to us) whether this equivalence is the same as the one we propose, or whether such traces can be manipulated explicitly as proof objects if desired.

In future work we may explore trace structures explicitly quotiented by causal equivalence, such as dependence graphs [17] or event structures [4]. We are also interested in extending braiding congruence to the full π-calculus structural congruence, and in understanding whether and how ideas from homotopy type theory [24], such as quotients or higher inductive types, could be applied to ease reasoning about or correct programming with π-calculus terms (modulo structural congruence) or traces (modulo causal equivalence).

Acknowledgements The authors were supported by the Air Force Office of Scientific Research, Air Force Material Command, USAF, under grant number FA8655-13-1-3006. The first author was also supported by UK EPSRC project *From Data Types to Session Types: A Basis for Concurrency and Distribution* (EP/K034413/1).

References

[1] David Baelde, Kaustuv Chaudhuri, Andrew Gacek, Dale Miller, Gopalan Nadathur, Alwen Tiu & Yuting Wang (2014): *Abella: A System for Reasoning about Relational Specifications. J. Formalized Reasoning* 7(2).

[2] Jesper Bengtson & Joachim Parrow (2009): *Formalising the pi-calculus using nominal logic. Logical Methods in Computer Science* 5(2:16).

[3] Michele Boreale & Davide Sangiorgi (1998): *A Fully Abstract Semantics for Causality in the π-Calculus. Acta Inf.* 35(5), pp. 353–400.

[4] Gérard Boudol & Ilaria Castellani (1989): *Permutation of transitions: An event structure semantics for CCS and SCCS.* In J.W. Bakker, W.-P. Roever & G. Rozenberg, editors: *Linear Time, Branching Time and Partial Order in Logics and Models for Concurrency*, *LNCS* 354, Springer, pp. 411–427, doi:10.1007/BFb0013028.

[5] N.G. de Bruijn (1972): *Lambda-Calculus Notation with Nameless Dummies: a Tool for Automatic Formula Manipulation with Application to the Church-Rosser Theorem. Indagationes Mathematicae* 34(5), pp. 381–392.

[6] Iliano Cervesato, Frank Pfenning, David Walker & Kevin Watkins (2002): *A Concurrent Logical Framework II: Examples and Applications.* Technical Report CMU-CS-02-102, Carnegie Mellon University.

[7] James Cheney & Roly Perera (2014): *An Analytical Survey of Provenance Sanitization.* In: *IPAW*, pp. 113–126.

[8] Ioana Cristescu, Jean Krivine & Daniele Varacca (2013): *A compositional semantics for the reversible pi-calculus.* In: *LICS*, pp. 388–397.

[9] Haskell B. Curry & Robert Feys (1958): *Combinatory Logic. Studies in Logic and the Foundations of Mathematics* 1, North-Holland, Amsterdam, Holland.

[10] Vincent Danos & Jean Krivine (2004): *Reversible Communicating Systems.* In Philippa Gardner & Nobuko Yoshida, editors: *CONCUR*, *LNCS* 3170, Springer, pp. 292–307, doi:10.1007/978-3-540-28644-8_19.

[11] Pierpaolo Degano & Corrado Priami (1999): *Non-Interleaving Semantics for Mobile Processes. Theor. Comput. Sci.* 216(1-2), pp. 237–270, doi:10.1016/S0304-3975(99)80003-6.

[12] Joëlle Despeyroux (2000): *A Higher-Order Specification of the pi-Calculus.* In: *IFIP TCS*, Springer-Verlag, pp. 425–439.

[13] Daniel Hirschkoff (1997): *A Full Formalisation of pi-Calculus Theory in the Calculus of Constructions*. In: *TPHOLs*, pp. 153–169, doi:10.1007/BFb0028392.

[14] Daniel Hirschkoff (1999): *Handling Substitutions Explicitly in the pi-Calculus*. In: *Proceedings of the Second International Workshop on Explicit Substitutions: Theory and Applications to Programs and Proofs*.

[15] Furio Honsell, Marino Miculan & Ivan Scagnetto (2001): *π-calculus in (Co)Inductive-type Theory*. *Theor. Comput. Sci.* 253(2), pp. 239–285, doi:10.1016/S0304-3975(00)00095-5.

[16] Jean-Jacques Lévy (1980): *Optimal reductions in the lambda-calculus*. In J. P. Seldin & J. R. Hindley, editors: *To H. B. Curry: Essays in Combinatory Logic, Lambda Calculus and Formalism*, Academic Press, pp. 159–191.

[17] A. Mazurkiewicz (1987): *Trace Theory*. In: *Advances in Petri Nets 1986, Part II on Petri Nets: Applications and Relationships to Other Models of Concurrency*, Springer-Verlag, pp. 279–324.

[18] Robin Milner (1999): *Communicating and mobile systems: the π calculus*. Cambridge University Press.

[19] Robin Milner, Joachim Parrow & David Walker (1992): *A Calculus of Mobile Processes, I and II*. *Inf. Comput.* 100(1), pp. 1–77, doi:10.1016/0890-5401(92)90009-5.

[20] Ulf Norell (2009): *Dependently Typed Programming in Agda*. In: *Advanced Functional Programming*, *LNCS* 5832, Springer, pp. 230–266.

[21] Dominic A. Orchard & Nobuko Yoshida (2015): *Using session types as an effect system*. In: *PLACES*.

[22] Roly Perera, Umut A. Acar, James Cheney & Paul Blain Levy (2012): *Functional Programs That Explain Their Work*. In: *ICFP*, ACM, pp. 365–376, doi:10.1145/2364527.2364579.

[23] Vaughan Pratt (2000): *Higher Dimensional Automata Revisited*. *Mathematical Structures in Computer Science* 10(4), pp. 525–548, doi:10.1017/S0960129500003169. Available at `http://dx.doi.org/10.1017/S0960129500003169`.

[24] The Univalent Foundations Program (2013): *Homotopy type theory: Univalent foundations of mathematics*. Technical Report, Institute for Advanced Study.

[25] Eugene W. Stark (1989): *Concurrent Transition Systems*. *Theoretical Computer Science* 64(3), pp. 221–269, doi:10.1016/0304-3975(89)90050-9.

[26] Alwen Tiu & Dale Miller (2010): *Proof Search Specifications of Bisimulation and Modal Logics for the π-calculus*. *ACM Trans. Comput. Logic* 11(2), pp. 13:1–13:35, doi:10.1145/1656242.1656248.

A Agda module structure

Figure 8 summarises the module structure of the Agda formalisation.

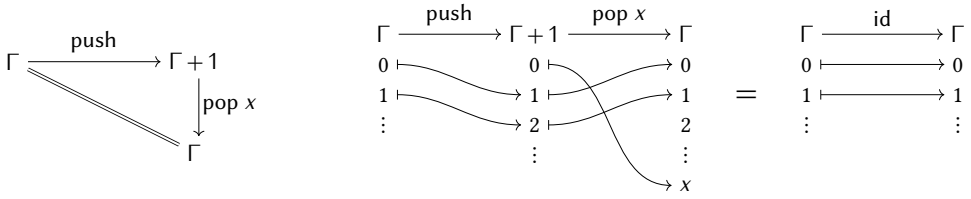
Utilities	
`Common`	Useful definitions not found in the Agda standard library
`SharedModules`	Common imports from standard library
Core modules	
`Action`	Actions a
`Action.Concur`	Concurrent actions $a \smile a'$; residuals a/a'
`Action.Concur.Action`	Residual of $a \smile a'$ after a''
`Action.Seq`	Action sequences $\widetilde{a}$
`Name`	Contexts Γ; names x
`Proc`	Processes P
`Ren`	Renamings $\rho : \Gamma \longrightarrow \Gamma'$
`StructuralCong.Proc`	Braiding congruence relation $\phi : P \cong P'$
`StructuralCong.Transition`	Residuals E/ϕ and ϕ/E
`Transition`	Transitions $E : P \xrightarrow{a} R$
`Transition.Concur`	Concurrent transitions $\chi : E \smile E'$; residuals E/E'
`Transition.Concur.Cofinal`	Cofinality braidings γ
`Transition.Concur.Cofinal.Transition`	Residuals E/γ and γ/E
`Transition.Concur.Transition`	Residual χ/E
`Transition.Seq`	Transition sequences
`Transition.Seq.Cofinal`	Residuals t/γ and γ/t; permutation equivalence $\alpha : t \simeq u$
Typical sub-modules	
`.Properties`	Additional properties relating to X
`.Ren`	Renaming lifted to X

Figure 8: Module overview

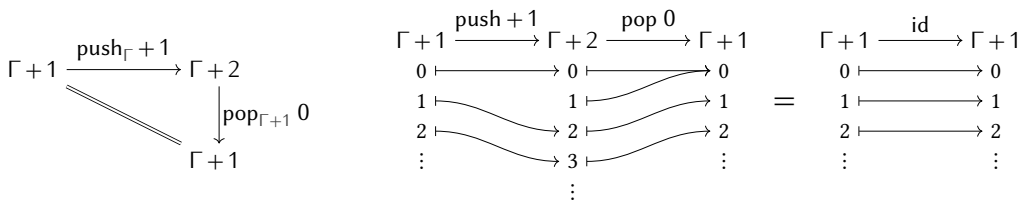
B Renaming lemmas

Each lemma asserts the commutativity of the diagram on the left; when a string diagram is also provided, it should be interpreted as an informal proof sketch.

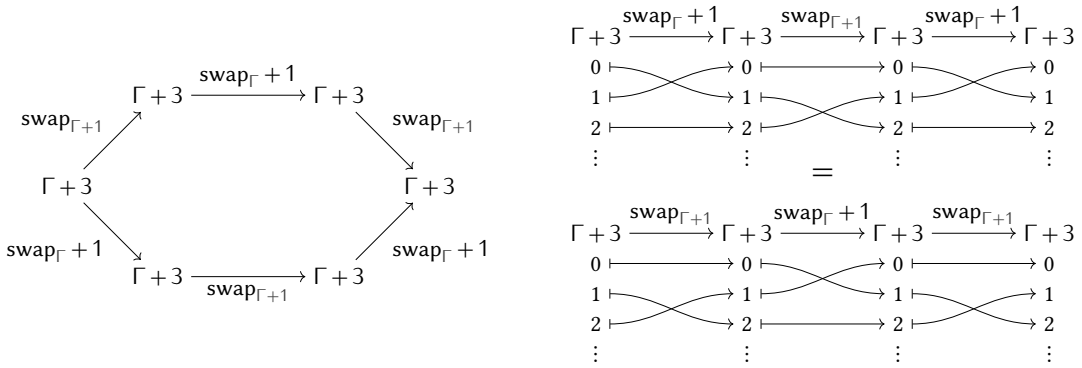
Lemma 1.



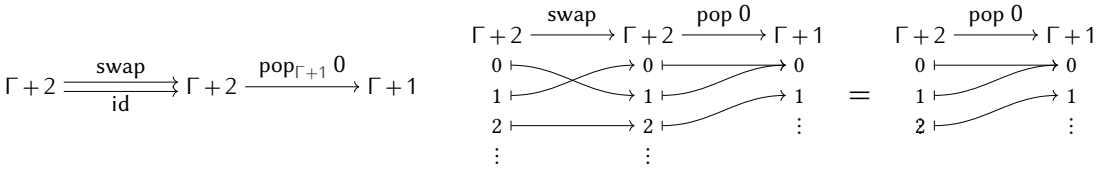
Lemma 2.



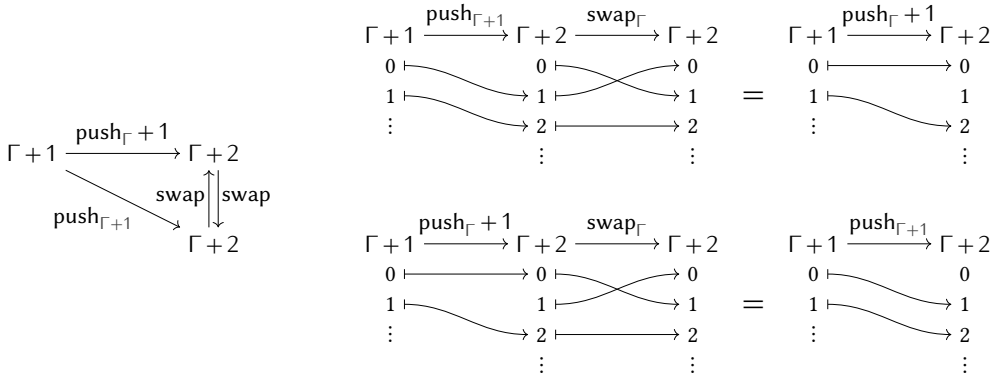
Lemma 3.



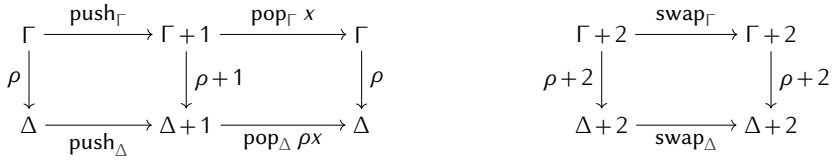
Lemma 4.



Lemma 5.



Lemmas 6, 7 and 8.



C Additional proofs

Proof of Lemma 9. By the following mutually recursive proofs-by-induction on the derivations. The various renaming lemmas needed to enable the induction hypothesis in each case are omitted.

$\rho^* E^c$

$$
\begin{aligned}
\rho^*(\overline{x}\langle y\rangle.P) &= \overline{\rho x}\langle \rho y\rangle.\rho^* P \\
\rho^*(E+F) &= \rho^* E + \rho^* F \\
\rho^*(P \mid^c F) &= \rho^* P \mid^{\rho^* c} \rho^* F \\
\rho^*(E^{\,c} \mid Q) &= \rho^* E^{\,\rho^* c} \mid \rho^* Q \\
\rho^*(E \mid^\tau_y F) &= \rho^* E \mid^\tau_{\rho^* y} \rho^* F \\
\rho^*(E \mid^\tau_\nu F) &= \rho^* E \mid^\tau_\nu \rho^* F \\
\rho^*(\nu^c E) &= \nu^{\rho^* c}(\rho+1)^* E \\
\rho^*(!E) &= !\rho^* E
\end{aligned}
$$

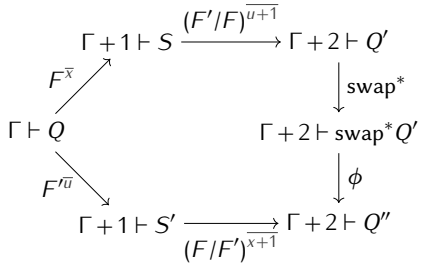
$\rho^* E^b$

$$
\begin{aligned}
\rho^*(\underline{x}.P) &= \underline{\rho x}.(\rho+1)^* P \\
\rho^*(E+F) &= \rho^* E + \rho^* F \\
\rho^*(P \mid^b F) &= \rho^* P \mid^{\rho^* b} \rho^* F \\
\rho^*(E^{\,b} \mid Q) &= \rho^* E^{\,\rho^* b} \mid \rho^* Q \\
\rho^*(\overline{\nu} E) &= \overline{\nu}(\rho+1)^* E \\
\rho^*(\nu^b E) &= \nu^{\rho^* b}(\rho+1)^* E \\
\rho^*(!E) &= !\rho^* E
\end{aligned}
$$

C.1 Additional illustrative cases of Theorem 1

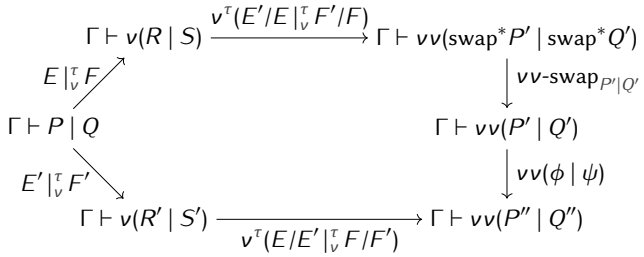
Example: permuting concurrent extrusions (different binders). First, note that the residuals of bound output transitions are not themselves necessarily bound. More specifically, the residuals of the output transition on $\overline{x}$ with the output on $\overline{z}$ is bound only if the outputs represent extrusions of different ν-binders. In this section we consider only the case when the concurrent extrusions are of different ν-binders.

In this case, each binder is unaffected by the extrusion of the other, and the residuals remain bound outputs, shifted into $\Gamma + 1$ as usual. The general form of such residuals is:



where ϕ ranges over braiding congruence. Then the residual is able to handle the inner extrusion, with the resulting τ action again propagated through the outer binder:

$$
\cdot \mid^\tau_\nu \cdot \;
\frac{
E \;\dfrac{\vdots}{\Gamma \vdash P \xrightarrow{\underline{x}} R}
\qquad
F \;\dfrac{\vdots}{\Gamma \vdash Q \xrightarrow{\overline{x}} S}
}{
\Gamma \vdash P \mid Q \xrightarrow{\tau} \nu(R \mid S)
}
$$



Example: permuting concurrent extrusions (same binder). Consider the process $\nu(\overline{x+1}\langle 0\rangle.P \mid \overline{z+1}\langle 0\rangle.Q)$, as described in Cristescu et al. [8]. There are two concurrent outputs, both of which try to extrude the top-level binder. Suppose we take the $\overline{x+1}\langle 0\rangle$ action first:

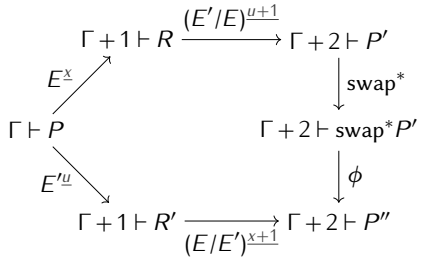
$$\overline{\nu}\cdot\ \frac{\cdot\ |\ \cdot\ \dfrac{\overline{x+1}\langle 0\rangle.P \xrightarrow{\overline{x+1}\langle 0\rangle} P}{\Gamma+1 \vdash \overline{x+1}\langle 0\rangle.P \mid \overline{z+1}\langle 0\rangle.Q \xrightarrow{\overline{x+1}\langle 0\rangle} \Gamma+1 \vdash P \mid \overline{z+1}\langle 0\rangle.Q}}{\Gamma \vdash \nu(\overline{x+1}\langle 0\rangle.P \mid \overline{z+1}\langle 0\rangle.Q) \xrightarrow{\overline{x}} \Gamma+1 \vdash P \mid \overline{z+1}\langle 0\rangle.Q}$$

If we then take the $\overline{z+1}\langle 0\rangle$ action, the enclosing ν-binder no longer exists, and so $\overline{z+1}\langle 0\rangle$ simply propagates as a non-bound action.

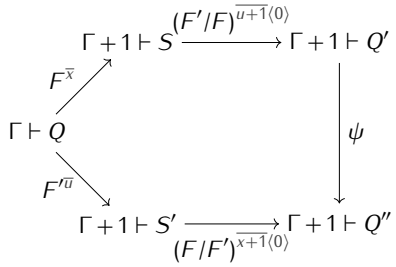
$$P\,|^{\overline{z+1}\langle 0\rangle}\cdot\ \frac{\Gamma+1 \vdash \overline{z+1}\langle 0\rangle.Q \xrightarrow{\overline{z+1}\langle 0\rangle} \Gamma+1 \vdash Q}{\Gamma+1 \vdash P \mid \overline{z+1}\langle 0\rangle.Q \xrightarrow{\overline{z+1}\langle 0\rangle} \Gamma+1 \vdash P \mid Q}$$

Example: permuting one extrusion-rendezvous with another. Now consider what happens when the extrusions from the previous example eventually rendezvous with a compatible input.

$$E\,|^{\tau}_{\nu} F : \Gamma \vdash P \mid Q \xrightarrow{\tau} \Gamma \vdash \nu(R \mid S)$$
$$E'\,|^{\tau}_{\nu} F' : \Gamma \vdash P \mid Q \xrightarrow{\tau} \Gamma \vdash \nu(R' \mid S')$$



When the extrusions are of the same ν-binder, and the residual outputs are not bound, then we have:



and the residual of one extrusion-handling after another is a plain communication, with the resulting τ action simply propagated through the second ν binder:

$$\begin{array}{ccc}
\Gamma \vdash \nu(R \mid S) & \xrightarrow{\nu^{\tau}(E'/E\,|^{\tau}_{0}\,F'/F)} & \Gamma \vdash \nu((\mathrm{pop}\ 0)^{*}\mathrm{swap}^{*}P' \mid Q') \\
{\scriptstyle E\,|^{\tau}_{\nu}\,F}\nearrow & & \| \\
\Gamma \vdash P \mid Q & & \nu(\alpha \mid Q') \\
{\scriptstyle E'\,|^{\tau}_{\nu}\,F'}\searrow & & \Gamma \vdash \nu((\mathrm{pop}\ 0)^{*}P' \mid Q') \\
 & & \downarrow{\scriptstyle \nu((\mathrm{pop}\ 0)^{*}\phi \mid \psi)} \\
\Gamma \vdash \nu(R' \mid S') & \xrightarrow[\nu^{\tau}(E/E'\,|^{\tau}_{0}\,F/F')]{} & \Gamma \vdash \nu((\mathrm{pop}\ 0)^{*}P'' \mid Q'')
\end{array}$$

Here α is the equality $(\mathrm{pop}\ 0) \circ \mathrm{swap} = \mathrm{pop}\ 0$ (Lemma 4) applied to P'.

Example: permuting bound actions propagating through a binder. Now suppose we have a process of the form νP which has two concurrent transitions propagating an input action through the ν binder:

$$\nu^{\underline{x}}\cdot\ \dfrac{E\ \dfrac{\vdots}{\Gamma+1 \vdash P \xrightarrow{\underline{x+1}} \Gamma+2 \vdash R}}{\Gamma \vdash \nu P \xrightarrow{\underline{x}} \Gamma+1 \vdash \nu(\mathrm{swap}^{*}R)} \qquad \nu^{\underline{u}}\cdot\ \dfrac{E'\ \dfrac{\vdots}{\Gamma+1 \vdash P \xrightarrow{\underline{u+1}} \Gamma+2 \vdash R'}}{\Gamma \vdash \nu P \xrightarrow{\underline{u}} \Gamma+1 \vdash \nu(\mathrm{swap}^{*}R')}$$

(The derivations are valid because both $\underline{x+1}$ and $\underline{z+1}$ are of the form $\mathrm{push}^{*}b$.) The residuals of E and E' with respect to each other have the form:

$$\begin{array}{ccc}
\Gamma+2 \vdash R & \xrightarrow{(E'/E)^{\underline{u+2}}} & \Gamma+3 \vdash P' \\
{\scriptstyle E^{\underline{x+1}}}\nearrow & & \downarrow{\scriptstyle \mathrm{swap}^{*}} \\
\Gamma+1 \vdash P & & \Gamma+3 \vdash \mathrm{swap}^{*}P' \\
{\scriptstyle E'^{\underline{u+1}}}\searrow & & \downarrow{\scriptstyle \phi} \\
\Gamma+2 \vdash R' & \xrightarrow[(E/E')^{\underline{x+2}}]{} & \Gamma+3 \vdash P''
\end{array}$$

We can use these residuals to define the following composite residual $(\nu^{\underline{u}}E')/(\nu^{\underline{x}}E)$:

$$\nu\cdot\ \dfrac{\mathrm{swap}^{*}\cdot\ \dfrac{E'/E\ \dfrac{\vdots}{\Gamma+2 \vdash R \xrightarrow{\underline{u+2}} \Gamma+3 \vdash P'}}{\Gamma+2 \vdash \mathrm{swap}^{*}R \xrightarrow{\underline{u+2}} \Gamma+3 \vdash (\mathrm{swap}+1)^{*}P'}}{\Gamma+1 \vdash \nu(\mathrm{swap}^{*}R) \xrightarrow{\underline{u+1}} \Gamma+2 \vdash \nu(\mathrm{swap}^{*}(\mathrm{swap}+1)^{*}P')}$$

noting that $\mathrm{swap}^{*}(\underline{u+2}) = \underline{u+2}$ by Lemma 8. The complementary residual $(\nu^{\underline{x}}E)/(\nu^{\underline{u}}E')$ is similar, with x instead of u and R' instead of R. It remains to show that the terminal states are swap-congruent:

$$\begin{array}{lr}
\quad \mathrm{swap}^{*}\nu(\mathrm{swap}^{*}(\mathrm{swap}+1)^{*}P') & \\
= \nu((\mathrm{swap}+1)^{*}\mathrm{swap}^{*}(\mathrm{swap}+1)^{*}P') & (\text{definition of } \cdot^{*}) \\
= \nu(\mathrm{swap}^{*}(\mathrm{swap}+1)^{*}\mathrm{swap}^{*}P') & (\text{Lemma 3}) \\
\cong \nu(\mathrm{swap}^{*}(\mathrm{swap}+1)^{*}P'') & (\nu(\mathrm{swap}^{*}(\mathrm{swap}+1)^{*}\phi))
\end{array}$$

Example: permuting extruding rendezvous and unhandled extrusion. Of course concurrent transitions are not always as symmetric as the ones we have seen. Here a name extrusion which has a successful rendezvous, resulting in a τ action, is concurrent with another which does not and which therefore propagates as a bound output:

$$P \mid^{\overline{u}}_{\nu} F : \Gamma \vdash P \mid Q \xrightarrow{\overline{u}} \Gamma + 1 \vdash \mathsf{push}^* P \mid S$$
$$E \mid^{\tau}_{\nu} F' : \Gamma \vdash P \mid Q \xrightarrow{\tau} \Gamma \vdash \nu(R \mid S')$$

As before, it matters whether the extrusions $F^{\overline{x}} \smile F'^{\overline{u}}$ are of the same or different binders.

Sub-case: extrusions of same binders. In this case, the residuals F'/F and F/F' become sends of index 0, the binder being extruded.

$$\cdot \mid^{\tau}_{0} \cdot \dfrac{\mathsf{push}^* \cdot \dfrac{E \dfrac{\vdots}{\Gamma \vdash P \xrightarrow{\overline{x}} \Gamma + 1 \vdash R}}{\Gamma + 1 \vdash \mathsf{push}^* P \xrightarrow{\overline{x+1}} \Gamma + 2 \vdash (\mathsf{push} + 1)^* R} \qquad F'/F \dfrac{\vdots}{\Gamma + 1 \vdash S \xrightarrow{\overline{x+1}\langle 0 \rangle} \Gamma + 1 \vdash Q'}}{\Gamma + 1 \vdash \mathsf{push}^* P \mid S \xrightarrow{\tau} \Gamma + 1 \vdash (\mathsf{pop}\ 0)^* (\mathsf{push} + 1)^* R \mid Q'}$$

For the other residual, we can derive:

$$\overline{\nu} \cdot \dfrac{R \mid^{\overline{u+1}\langle 0 \rangle} \cdot \dfrac{F/F' \dfrac{\vdots}{\Gamma + 1 \vdash S' \xrightarrow{\overline{u+1}\langle 0 \rangle} \Gamma + 1 \vdash Q''}}{\Gamma + 1 \vdash R \mid S' \xrightarrow{\overline{u+1}\langle 0 \rangle} \Gamma + 1 \vdash R \mid Q''}}{\Gamma \vdash \nu(R \mid S') \xrightarrow{\overline{u}} \Gamma + 1 \vdash R \mid Q''}$$

with $Q' \cong Q''$, and noting that pop 0 retracts push $+ 1$ (Lemma 2 below).

Sub-case: extrusions of different binders. In this case the residuals F'/F and F/F' remain bound outputs. Then, with the $\mathsf{push}^* E$ derivation as before, we can derive:

$$\mathsf{push}^* E \mid^{\tau}_{\nu} \cdot \dfrac{F'/F \dfrac{\vdots}{\Gamma + 1 \vdash S \xrightarrow{\overline{x+1}} \Gamma + 2 \vdash Q'}}{\Gamma + 1 \vdash \mathsf{push}^* P \mid S \xrightarrow{\tau} \Gamma + 1 \vdash \nu((\mathsf{push} + 1)^* R \mid Q')}$$

and for the other residual:

$$\nu^{\overline{u}} \cdot \dfrac{R \mid^{\overline{u+1}} \cdot \dfrac{F/F' \dfrac{\vdots}{\Gamma + 1 \vdash S' \xrightarrow{\overline{u+1}} \Gamma + 2 \vdash Q''}}{\Gamma + 1 \vdash R \mid S' \xrightarrow{\overline{u+1}} \Gamma + 2 \vdash \mathsf{push}^* R \mid Q''}}{\Gamma \vdash \nu(R \mid S') \xrightarrow{\overline{u}} \Gamma + 1 \vdash \nu(\mathsf{swap}^* \mathsf{push}^* R \mid \mathsf{swap}^* Q'')}$$

with $\mathsf{swap}^* Q' \cong Q''$. It remains to establish a $\cong$-path between the two terminal processes. We have $Q' \cong \mathsf{swap}^* Q''$ by functionality and involutivity of swap, and push $+ 1 =$ swap $\circ$ push by Lemma 5 and then the rest follows by reflexivity and congruence.

C.2 Cofinality for Theorem 4

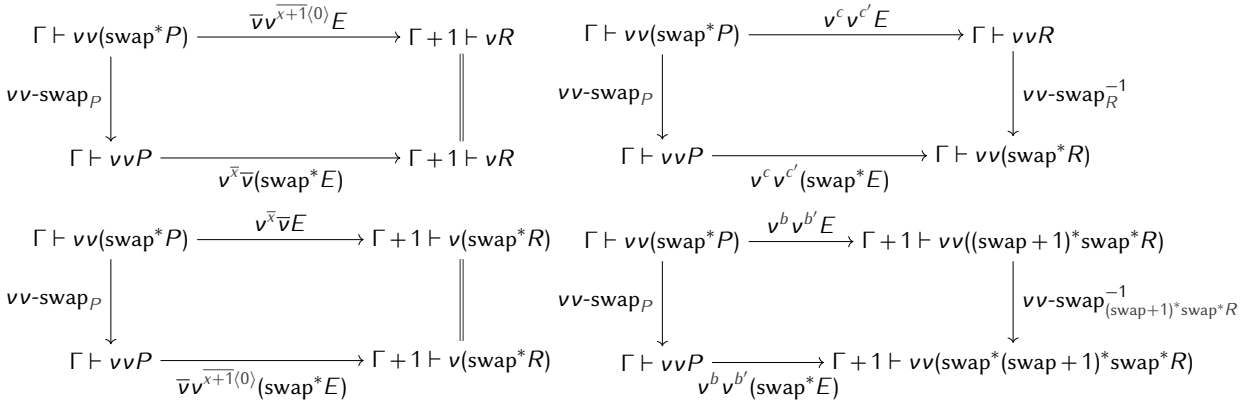


Figure 9: Cofinality of ϕ/E and E/ϕ in the $\nu\nu$-swap cases

Figure 9 illustrates cofinality for the $\nu\nu$-swap cases, omitting the renaming lemmas used as type-level coercions. The $\nu\nu$-swap^{-1} cases are symmetric.