Models for Name-Passing Processes: Interleaving and Causal (Extended Abstract)

Gian Luca Cattani* and Peter Sewell†
Computer Laboratory, University of Cambridge, England
{Luca.Cattani,Peter.Sewell}@cl.cam.ac.uk

Abstract

We study syntax-free models for name-passing processes. For interleaving semantics, we identify the indexing structure required of an early labelled transition system to support the usual π-calculus operations, defining Indexed Labelled Transition Systems. *For non-interleaving causal semantics we define* Indexed Labelled Asynchronous Transition Systems, *smoothly generalizing both our interleaving model and the standard Asynchronous Transition Systems model for CCS-like calculi. In each case we relate a denotational semantics to an operational view, for bisimulation and causal bisimulation respectively. This is a first step towards a uniform understanding of the semantics and operations of name-passing calculi.*

1. Introduction

The study of concurrency has involved rich interplay between model-theoretic and syntactic approaches. The first takes a notion of behaviour – perhaps defined as some class of automata or labelled transition systems – as primary; the second focusses on some particular signature of process terms, perhaps giving it only an axiomatic semantics. It is now common to take an intermediate approach: to fix a signature of process terms and equip it with an operational semantics defining behaviour (e.g. transition relations) over those terms. This has been followed for almost all work on π-calculi, beginning with [MPW92], in which an operational semantics defines transition relations with particular labels over π-terms. By contrast, in this paper we study purely model-theoretic notions of behaviour for π-calculi, with definitions that do not involve process syntax, to support the uniform development of metatheory for a range of calculi and semantics. For interleaving semantics we introduce *Indexed Labelled Transition Systems* with data specifying how transitions change under renaming – thus picking out the essential structure of a π early transition relation that is required for defining the normal operations and equivalences over π-terms. For non-interleaving causal semantics, we define *Indexed Labelled Asynchronous Transition Systems*, smoothly generalizing both our interleaving model and the standard Asynchronous Transition Systems model for CCS-like calculi [Bed88, Shi85, WN95]. In each case we give a denotational semantics of a π-calculus; we prove the operational early and causal bisimulations [San93, BS98] coincide with model-theoretic notions. In the full version of this extended abstract [CS00], we also investigate the properties of and relationship between categories of the two models, and give the omitted proofs. This is a first step towards a uniform understanding of the semantics and operations of name-passing calculi.

Interleaving The standard notion of labelled transition system (LTS) for calculi without value-passing is straightforward. For example, given a set N of names (ranged over by $a, b, \ldots$) the CCS fragment

$$P ::= 0 \;\big|\; \overline{a}.P \;\big|\; a.P \;\big|\; P \mid Q \;\big|\; (\boldsymbol{\nu} c)P$$

can be given semantics in terms of LTSs

$$\langle S, \longrightarrow, i\rangle$$

where S is a set of states, $\longrightarrow \subseteq S \times \mathcal{L} \times S$ is a transition relation with labels $\mathcal{L} = \{\tau, a, \overline{a}, b, \overline{b}, \ldots\}$, and $i \in S$ is the initial state. Introducing value-passing, however, makes the situation more complex – particularly with scope extrusion. Consider the π-calculus fragment below, in which the 'c' in the input $bc.P$ and restriction $(\boldsymbol{\nu} c)P$ bind in the process P.

$$P ::= 0 \;\big|\; \overline{a}d.P \;\big|\; bc.P \;\big|\; P \mid Q \;\big|\; (\boldsymbol{\nu} c)P$$

Defining the behaviour of $bc.P$ involves substitution. For example, the communication of a free name

$$\overline{a}d.P \mid ac.Q \xrightarrow{\tau} P \mid \{^d/_c\}Q$$

*supported by EPSRC grant GR/L62290: Calculi for Interactive Systems: Theory and Experiment

†supported by a Royal Society University Research Fellowship

is inferred in the 'early' semantics of [MPW93, San93] with the rules below.

$$\textsc{Out}\ \frac{}{\overline{a}d.P \xrightarrow{\overline{a}d} P} \qquad \textsc{In}\ \frac{}{ac.Q \xrightarrow{ad} \{^d/_c\}Q}$$

$$\textsc{Com}\ \frac{P \xrightarrow{\overline{a}d} P' \quad Q \xrightarrow{ad} Q'}{P \mid Q \xrightarrow{\tau} P' \mid Q'}$$

Note that d might or might not be in the free names of Q. Moreover, unlike in CCS, π-calculus τ-transitions can also involve scope extrusion:

$$((\boldsymbol{\nu} d)\overline{a}d.P) \mid ac.Q \xrightarrow{\tau} (\boldsymbol{\nu} d)(P \mid \{^d/_c\}Q) \quad \text{if } d \notin \text{fn}(Q)$$

To define the τ-transitions of $P \mid Q$ compositionally, in terms of the transitions of P and Q, the semantics must distinguish between outputs of free and bound names, by taking transitions with labels $\overline{a}d$ and $\overline{a}(d)$ respectively. The τ-transition above can be inferred with the rules:

$$\textsc{Open}\ \frac{P \xrightarrow{\overline{a}d} P' \quad d \neq a}{(\boldsymbol{\nu} d)P \xrightarrow{\overline{a}(d)} P'}$$

$$\textsc{Close}\ \frac{P \xrightarrow{\overline{a}(d)} P' \quad Q \xrightarrow{ad} Q' \quad d \notin \text{fn}(Q)}{P \mid Q \xrightarrow{\tau} (\boldsymbol{\nu} d)(P' \mid Q')}$$

The full semantics requires also the rules

$$\textsc{Res}\ \frac{P \xrightarrow{\ell} P' \quad d \notin \text{fn}(\ell)}{(\boldsymbol{\nu} d)P \xrightarrow{\ell} (\boldsymbol{\nu} d)P'}$$

$$\textsc{Par}\ \frac{P \xrightarrow{\ell} P' \quad \text{bn}(\ell) \cap \text{fn}(Q) = \emptyset}{P \mid Q \xrightarrow{\ell} P' \mid Q}$$

(in which $\text{bn}(\overline{a}(d)) = \{d\}$, and $\text{bn}(\ell) = \emptyset$ for labels of other forms) for restricted transitions that do not involve scope extrusion and for parallel.

These SOS rules involve subtle conditions on the free names of process terms (relating them to names in labels), in addition to name substitution on process terms. To give a syntax-free notion of LTS that has enough structure to define the operations we must therefore consider states not simply to be elements of an arbitrary set but of a set indexed by finite sets of names – the 'free' names of the states – and add data specifying how states change under renaming. In Section 3 we will define an Indexed Labelled Transition System (or $\mathcal{N}$-LTS) to have data

$$\langle S : \mathcal{N} \to \mathbf{Set},\ \longrightarrow,\ \langle I, i \rangle\rangle$$

where S is a functor from an indexing category $\mathcal{N}$ of name-sets and renamings into $\mathbf{Set}$ (the category of all sets and functions), giving the set of states above each name-set; the transition relation is over the coproduct $\coprod_{A \in |\mathcal{N}|} S(A)$; and the initial state $\langle I, i \rangle$ is an element of that coproduct. Axioms must be imposed, enforcing:

1. the name-sets of the endpoints of a transition must be related to each other and to the label;
2. input transitions occur in families related by renaming of the result states;
3. (a) transitions are preserved by injective renaming, both of the names of states and of new names in labels;
   (b) inputs of new names above a name-set give rise to inputs of old names above larger name-sets; and
4. the transitions of an injective renaming of a state are determined by the transitions of the state.

We give the precise definition of $\mathcal{N}$-LTS in Section 3, following a description of the π-calculus we are using in Section 2. Many variant definitions of $\mathcal{N}$-LTS are possible; we discuss the alternatives in Section 4. In Section 5 we define constructions over $\mathcal{N}$-LTSs, giving a denotational semantics, and relate bisimulation over $\mathcal{N}$-LTSs with the bisimulation defined using the operational semantics.

Non-Interleaving models for process calculi have been much studied; they can support model-checking techniques that mitigate the state-explosion problem, and strong proof techniques. They are also required in cases where the desired properties of systems are most naturally stated in terms of causality or locality. Here again there are model-theoretic and syntactic approaches – the first is surveyed in [WN95]; the second is represented by various annotated operational semantics, e.g. [DDNM88a, BC88, DD89, Kie94, WN95]. The two seem to have been carried out almost independently – to our knowledge, the only works to make precise connections are [DDNM88a, BC88, WN95]. Moreover, only the syntactic approach has been developed to address name-passing, in the annotated operational models of [BS98, DP99]. There is also work that does not fit this categorisation, having both syntactic and model-theoretic aspects, with Petri nets and graph rewriting [BG95, MP95].

Our goal in the second half of this paper is to develop the model-theoretic approach, and to make precise connections to the annotated operational notions. We develop a simple syntax-free non-interleaving model for name-passing that generalises both our interleaving model and the standard Asynchronous Transition Systems model for calculi without name-passing [Bed88, Shi85, WN95]. This is precisely related to causal bisimulation [BS98].

In CCS causal dependency arises from prefixing – in the behaviour of the process $\overline{x}.\overline{y}.0$ the y output causally depends on the x output. In π-calculus, name-binding introduces new dependencies, as thoroughly discussed in [DP99]. Transitions occurring in different parallel components of a process term, naively regarded as independent, may be forced to occur in a fixed order. For example, in the process $(\boldsymbol{\nu} y)(\overline{x}y \mid \overline{y}z)$ the transition $\overline{y}z$ can be observed only after $\overline{x}y$ – before this occurs the new-bound channel is not known to the environment. The two transitions of $(\boldsymbol{\nu} y)(\overline{x}y \mid \overline{z}y)$ are independent, however, despite the fact that the first to occur will be an output of a new name and the second will not. Further, an input of a previously-extruded name, e.g. $(\boldsymbol{\nu} y)(\overline{x}y \mid xw.0) \xrightarrow{\overline{x}y}\xrightarrow{xy} 0$, or output of a previously-input new name, e.g. $xw.\overline{x}w \xrightarrow{xy}\xrightarrow{\overline{x}y} 0$ (where y is new) involves dependency. Moreover, one can choose whether or not to distinguish between the prefix and name dependency, e.g. whether to identify $(\boldsymbol{\nu} y)(\overline{x}y.\overline{y}z)$ and $(\boldsymbol{\nu} y)(\overline{x}y \mid \overline{y}z)$.

In Section 6 we define a relation of name-dependency between two labels (wrt. a name-set), and then an Indexed Labelled Asynchronous Transition System (or $\mathcal{N}$-LATS) to have data

$$\langle S : \mathcal{N} \to \mathbf{Set},\ \longrightarrow,\ \langle I, i\rangle,\ E,\ \mathcal{I}\rangle$$

where now transitions are annotated by elements of a set E of *events* and $\mathcal{I} \subseteq E \times E$ is an *independence relation* between events. We impose axioms requiring that one obtains an Indexed LTS when considering each $e \in E$ separately, and (roughly), that independent transitions can be permuted. As one would expect, name dependency is involved in the relationship between the transition and independence relations. We discuss how the constructions of Section 5 can be extended to $\mathcal{N}$-LATS, define history-preserving bisimulation and a name-dependency aware variant (respectively distinguishing and identifying the example two processes above), and prove correspondence results.

In [CS00] an abstract study of the structures defined in this paper is initiated. Categories $\mathcal{N}\text{-}\mathbf{LTS}_I$ of Indexed LTS and $\mathcal{N}\text{-}\mathbf{LATS}_I$ of Indexed LATS (each for initial name-set I) are defined. Their properties and mutual relationship are studied, as the first step towards an abstract understanding of the equivalences and constructions involved in the semantics of π-like process languages. Space limitations prevent us from presenting such results here.

Further Motivation, Future Directions and Related Work Viewing models categorically has proven useful in study of the interleaving/non-interleaving and linear-time/branching-time distinctions [SNW96]. Moreover, the categorical study of process calculi gives the possibility of obtaining general congruence results: in [WN95] categorical models of CCS-like processes are axiomatised and in [JNW96] an abstract model-theoretic notion of bisimulation is introduced (via open maps); in [CW97, CW99] these two are combined to give abstract congruence results for strong bisimulation over a wide range of models. It is our hope that the present work serves as a first step towards similar results for π-calculus-like process languages. In particular, we would like a categorical understanding of our operations for the two models, related by the results presented in [CS00].

Among earlier models of π-processes, the name passing synchronisation trees of [Hon99] and presheaves of [CSW97] are the closest to our $\mathcal{N}$-LTS, though they employ a slightly different indexing structure (*cf.* Section 4). In [CSW97] the models are defined using domain theoretical techniques similar to those employed in [Sta96, FMS96], as the solutions to semantic equations. By contrast here we take a more concrete approach, with several advantages. Firstly, it is easy to conceive of minor modifications to our definitions which adapt them to closely reflect paradigms such as the asynchronous π-calculus [Bou92, HT91]. In particular it should be quite straightforward to adapt the axioms of [Sel97] to our models. It should also be easy to address the πI-calculus [San96a], in which only new names are communicated (though this can also be done domain-theoretically). Secondly, it supports a direct definition of weak bisimulation, something the domain model lacks and the presheaf model can, as far as we know, only achieve indirectly by means of a saturation construction [FCW99].

It is also worth noticing that while the domain models are tailored for late bisimulation, our focus here is on early semantics, both to obtain a simpler notion of transition system, and because we have found the early style suits work on concurrent language semantics and on secure encapsulation [Sew97, SV99a, SV99b, Sew00]. Presheaf models exist for both early and late notions [Cat99]. Moreover we should add that, in constrast to [Sta96, FMS96] (which have full-abstraction results wrt. strong bisimulation), we focus on intensional models, over which a number of equivalences can be defined (though we give results only for bisimulation). The literature contains also testing-based models [Hen96, BDN95]. The precise relationships with these and other models defined in the literature e.g., [MP98, MP95, BG95, JJ95] requires further work.

In [CS00] we begin the study of three applications. Firstly, we believe our structures may form a useful basis for π-calculus interleaving and partial-order model checking, via notions of finitely-generable $\mathcal{N}$-LTS and $\mathcal{N}$-LATS. We define the former precisely. Secondly, $\mathcal{N}$-LTS (extended to allow communication of tuples and encrypted values) may provide a basis for proofs and model-checking of cryptographic protocols. In particular, it would support direct (i.e. not via a process calculus syntax) definitions of behaviours, as in [Pau98], while still allowing composition of these behaviours. Thirdly, developing work on secure encapsulation [SV99a, SV99b], quantifying over elements of the model, rather than over syntactic processes, would allow stronger security properties to be stated. We state conjectures relating the coloured π-calculus semantics – an approximate but simple notion of causality used there to state security properties – to $\mathcal{N}$-LATS. We also wonder what the relationships are with the recent [GP99, FPT99, Hof99], where similar indexing structure is used in a λ-calculus setting.

Finally, notice that in this paper we introduce transition systems with indexed sets of states, but not indexed sets of transitions. This is because, as remarked above, when moving from a name-set to a larger one, transitions labelled with inputs of new names in the former give rise to input transitions of both new and old names in the latter – the correspondence between transitions is not functional, even for injective renamings. We have begun to consider, with Hyland, more sophisticated indexing structures which allow transitions as well as states to be indexed; the pay-off for the extra complication being e.g. the possibility of using the notion of internal category to formally relate our Indexed Transition Systems with the standard ones.

2. Background on the π-Calculus

Many variant π-calculi have been studied in the literature since the original was introduced in [MPW92]. Here, to show the wide applicability of our models, we take a rich set of primitives including summation, matching, mismatching and synchronous output. For notational simplicity, however, we treat only a monadic untyped calculus without basic values; for lack of space also omitting replication. These could be easily added.

Syntax We take an infinite set N of *names* of channels, ranged over by a, b etc. The *process terms* are then those defined by the grammar

$$\begin{array}{rcl} P, Q & ::= & 0 \mid P \mid Q \mid P + Q \mid \tau.P \mid \overline{a}d.P \mid ac.P \mid !P \\ & & \mid (\boldsymbol{\nu} c)P \mid [a = b]P \mid [a \neq b]P\ . \end{array}$$

Here the c in the input $bc.P$ and restriction $(\boldsymbol{\nu} c)P$ bind in the process P; we work up to alpha renaming of bound names. We write $\mathrm{fn}(P)$ for the set of free names of P, and $\{a/b\}P$ for the process term obtained from P by replacing all free occurrences of b by a.

Operational semantics We equip the calculus with a mild presentational variant, explicitly-indexed, of the early labelled transition semantics of [San93, MPW93], in which transitions are given for processes with respect to explicit supersets of their free name sets. This style simplifies the SOS rules, allowing sideconditions in PAR and CLOSE (here coalesced with COM) to be removed, gives a simple notion of trace, and supports subtype systems; it has been useful for work on concurrent language semantics and on secure encapsulation [Sew97, SV99a, SV99b]. The labelled transition relation has the form

$$A \vdash P \xrightarrow{\ell} Q$$

where A is a finite set of names and $\mathrm{fn}(P) \subseteq A$; it should be read as 'in a state where the names A may be known by process P and by its environment, the process P can do ℓ to become Q. The name-set associated with Q is then $A \cup \mathrm{fn}(\ell)$. The labels Lab are $\{\tau\} \cup \{\, \overline{x}y \mid x, y \in N\,\} \cup \{\, xy \mid x, y \in N\,\}$. Note that we now have only one form of output label – a transition $A \vdash P \xrightarrow{\overline{x}v} Q$ is an output of a new name iff $v \notin A$. The transition relation is defined as the smallest relation satisfying the rules in Figure 1. The free names of a label are $\mathrm{fn}(\tau) = \{\}$, $\mathrm{fn}(\overline{x}v) = \mathrm{fn}(xv) = \{x, v\}$. We write A, x for $A \cup \{x\}$ where x is assumed not to be in A. If $A = \emptyset$ then $(\boldsymbol{\nu} A)P$ denotes P.

Note that the set of free names of a process can grow along transitions, for example $\{a\} \vdash (\boldsymbol{\nu} d)\overline{a}d.\overline{a}d \xrightarrow{\overline{a}d} \overline{a}d$, and that the rules depend in an essential way on alpha-conversion – the process $R = (\boldsymbol{\nu} d)\overline{a}d$ must be able to perform a bound output with label $\overline{a}\hat{d}$ for any $\hat{d} \neq a$; derivations of such transitions require use of the alpha-equivalence $R = (\boldsymbol{\nu}\hat{d})\overline{a}\hat{d}$. Note also that the SOS rules do not involve any structural congruence.

Example Properties We illustrate the SOS with some example transitions and properties – these will be special cases of the axioms imposed on $\mathcal{N}$-LTS in Section 3.

1. If $A \vdash P \xrightarrow{\overline{x}z} Q$ then $x \in A$. We might have z new, i.e. $z \in A$ or not, i.e. $z \notin A$. In either case, Q has free names contained in $A \cup \{x, z\}$. The same holds for input transitions.

2. A transition $A \vdash P \xrightarrow{xz} Q$ must arise from an input prefix in P, which must therefore be able to input any other name (new or old). Moreover, the resulting states can all be obtained by substitution from the resulting state after a new name is input.

3. (a) If $A \vdash P \xrightarrow{\overline{x}z} Q$ and $z \in A$ then for any injective substitution, say $f : A \rightarrow_{\mathrm{inj}} B$, there is a transition $B \vdash fP \xrightarrow{\overline{fx}fz} fQ$. For output of a new name, i.e. $z \notin A$, the value z can also be renamed to any $\hat{z} \notin B$, giving $B \vdash fP \xrightarrow{\overline{fx}\hat{z}} (f + [\hat{z}/z])Q$. The same holds for input transitions.
   (b) A derivation of an input $A \vdash P \xrightarrow{xz} Q$ of a new name $z \notin A$ is preserved by extending the name-set – so P above (A, z) has an input of an old name $A, z \vdash P \xrightarrow{xz} Q$.

4. Non-injective renaming can enable and (with mismatch) disable transitions, but the behaviour of an injective renaming of P is determined by that of P.

$$\textsc{Out}\ \frac{}{A \vdash \overline{x}v.P \xrightarrow{\overline{x}v} P} \qquad \textsc{Par}\ \frac{A \vdash P \xrightarrow{\ell} P'}{A \vdash P \mid Q \xrightarrow{\ell} P' \mid Q} \qquad \textsc{Res}\ \frac{A, x \vdash P \xrightarrow{\ell} P' \quad x \notin \mathrm{fn}(\ell)}{A \vdash (\boldsymbol{\nu} x)P \xrightarrow{\ell} (\boldsymbol{\nu} x)P'}$$

$$\textsc{In}\ \frac{}{A \vdash xp.P \xrightarrow{xv} \{^v/_p\}P} \qquad \textsc{Com}\ \frac{A \vdash P \xrightarrow{\overline{x}v} P' \quad A \vdash Q \xrightarrow{xv} Q'}{A \vdash P \mid Q \xrightarrow{\tau} (\boldsymbol{\nu}\{v\} \setminus A)(P' \mid Q')} \qquad \textsc{Open}\ \frac{A, x \vdash P \xrightarrow{\overline{y}x} P' \quad y \neq x}{A \vdash (\boldsymbol{\nu} x)P \xrightarrow{\overline{y}x} P'}$$

$$\textsc{Sum}\ \frac{A \vdash P \xrightarrow{\ell} P'}{A \vdash P + Q \xrightarrow{\ell} P'} \qquad \textsc{Match}\ \frac{A \vdash P \xrightarrow{\ell} P'}{A \vdash [x = x]P \xrightarrow{\ell} P'} \qquad \textsc{Mismatch}\ \frac{A \vdash P \xrightarrow{\ell} P' \quad x \neq y}{A \vdash [x \neq y]P \xrightarrow{\ell} P'}$$

In all rules with conclusion of the form $A \vdash P \xrightarrow{\ell} Q$ there is an implicit side condition $\mathrm{fn}(P) \subseteq A$. The rule for $\tau.P$ and symmetric versions of PAR, COM and SUM are elided.

Figure 1. π operational semantics

Operational Equivalences The normal notion of early bisimulation can be easily adapted to the explicitly-indexed setting. Take *bisimulation* $\dot\sim$ to be the largest family of relations indexed by finite sets of names such that each $\dot\sim_A$ is a symmetric relation over $\{ P \mid \mathrm{fn}(P) \subseteq A \}$ and for all $P \dot\sim_A Q$,

- if $A \vdash P \xrightarrow{\ell} P'$ then $\exists Q' \ .\ A \vdash Q \xrightarrow{\ell} Q' \wedge P' \dot\sim_{A \cup \mathrm{fn}(\ell)} Q'$.

We do not develop other equivalences in this paper, but linear-time notions can also be defined straightforwardly. For example, for partial traces write

$$A_1 \vdash P_1 \xrightarrow{\ell_1} \ldots \xrightarrow{\ell_n} P_{n+1}$$

to mean $\exists P_2, \ldots, P_n, A_2, \ldots, A_n \ .\ \forall i \in 1..n \ .\ A_{i+1} = A_i \cup \mathrm{fn}(\ell_i) \wedge A_i \vdash P_i \xrightarrow{\ell_i} P_{i+1}$. If $\mathrm{fn}(P) \subseteq A$ then the partial A-traces of P are simply $\{ \ell_1 .. \ell_n \mid \exists P' \ .\ A \vdash P \xrightarrow{\ell_1} \ldots \xrightarrow{\ell_n} P' \}$.

3. $\mathcal{N}$-LTS

In this section we introduce Indexed Labelled Transition Systems. To account for name substitution of π-terms, we take an indexing structure of name-sets and renaming functions on the set of states. We then axiomatize the key properties of the transition relation with respect to this indexing structure. We have also considered other choices of indexing structure, as briefly discussed in the next section.

Definition 3.1 *Take $\mathcal{N}$ to be the category with objects finite subsets of N and arrows functions $f : A \to B$ between them.*

NOTATION: If $f : A \to B$ and $g : A' \to B'$ are two functions we write $f + g$ for the obvious function $A \uplus A' \to B \uplus B'$. If $f : A \to B$ and $g : A' \to B$ we write $[f, g]$ for the obvious copairing function $A \uplus A' \to B$. Given a function $f : A \to B$ and two names $x \notin A$ and $y \in B$, write $[f, [y/x]]$ for the obvious *copairing* function $A, x \to B$.

If $S : \mathcal{N} \to \mathbf{Set}$ is a functor and if $\coprod_{A \in |\mathcal{N}|} S(A)$ is the disjoint union of the sets $S(A)$ for objects A of $\mathcal{N}$, write $\langle A, s \rangle$ for the element $s \in S(A)$ as an element of the disjoint union and S for the set $\coprod_{A \in |\mathcal{N}|} S(A)$ itself. If $\longrightarrow \subseteq \mathsf{S} \times \mathrm{Lab} \times \mathsf{S}$ is a (transition) relation, we will write $A \vdash \mathsf{s} \xrightarrow{\ell} \mathsf{t}$ to mean that there exists an $s \in S(A)$, a set B and a $t \in S(B)$ such that $\mathsf{s} = \langle A, s \rangle$, $\mathsf{t} = \langle B, t \rangle$ and $\mathsf{s} \xrightarrow{\ell} \mathsf{t}$. Sometimes we want to make explicit the existence of B and write $A \vdash \mathsf{s} \xrightarrow{\ell} \mathsf{t} \dashv B$ to this purpose. Also if $f : A \to B$ is a function, write $f\mathsf{s}$ for $\langle B, S(f)(s) \rangle$.

Definition 3.2 *For any label $\ell \in$ Lab, define the* channel names *of ℓ, $\mathrm{chan}(\ell)$ and the* value names *of ℓ, $\mathrm{val}(\ell)$as follows:*

$$\begin{array}{rclrcl} \mathrm{chan}(\tau) & = & \emptyset & \mathrm{val}(\tau) & = & \emptyset \\ \mathrm{chan}(\overline{x}y) & = & \{x\} & \mathrm{val}(\overline{x}y) & = & \{y\} \\ \mathrm{chan}(xy) & = & \{x\} & \mathrm{val}(xy) & = & \{y\} \end{array}$$

Definition 3.3 *Define an Indexed Labelled Transition System ($\mathcal{N}$-LTS) to be a structure*

$$T = \langle S : \mathcal{N} \to \mathbf{Set},\ \longrightarrow,\ \langle I, i \rangle \rangle$$

where $\langle I, i \rangle \in \mathsf{S}$, $\longrightarrow \subseteq \mathsf{S} \times \mathrm{Lab} \times \mathsf{S}$ and the following conditions hold.

1. *(Naming)* $A \vdash \mathsf{s} \xrightarrow{\ell} \mathsf{t} \dashv B \implies \mathrm{chan}(\ell) \subseteq A \wedge B = A \cup \mathrm{fn}(\ell)$
2. (a) *(Input – new)*$A \vdash \mathsf{s} \xrightarrow{xy} \mathsf{t} \dashv A, y \implies \forall z \in A.\ A \vdash \mathsf{s} \xrightarrow{xz} [1_A, [z/y]]\mathsf{t}$
   (b) *(Input – old)*$A \vdash \mathsf{s} \xrightarrow{xy} \mathsf{t} \dashv A \implies \forall z \notin A \exists \mathsf{t_z}.\ A \vdash \mathsf{s} \xrightarrow{xz} \mathsf{t_z} \dashv A, z \wedge \mathsf{t} = [1_A, [y/z]]\mathsf{t_z}$
3. (a) *(Injective substitution)*
   For $f : A \to_{inj} B,\ A \vdash \mathsf{s} \xrightarrow{\ell} \mathsf{t} \wedge g : (\mathrm{fn}(\ell) \setminus A) \to_{bij} \hat{B} \wedge \hat{B} \cap B = \emptyset \implies f\mathsf{s} \xrightarrow{(f+g)\ell} (f+g)\mathsf{t}$

(b) *(Shifting)*
$A \vdash \mathsf{s} \xrightarrow{xy} \mathsf{t} \dashv A, y \implies A, y \vdash \iota\mathsf{s} \xrightarrow{xy} \mathsf{t}$, *where* $\iota : A \hookrightarrow A, y$ *is the set inclusion function*

4. *For* $f : A \to_{inj} B$, *if* $f\mathsf{s} \xrightarrow{\ell'} \mathsf{t}'$ *then (at least) one of the following two cases applies*

   (a) *there exist* ℓ, t, $g : \mathrm{fn}(\ell) \setminus A \to_{bij} \hat{B}$ *such that* $\hat{B} \cap B = \emptyset$ *and* $\ell' = (f + g)(\ell)$ *and* $\mathsf{s} \xrightarrow{\ell} \mathsf{t}$ *and* $\mathsf{t}' = (f + g)\mathsf{t}$

   (b) *there exist* $x \in A$, $y \notin A$, $z \in B$ *and* t *such that* $\ell' = f(x)z$ *and* $A \vdash \mathsf{s} \xrightarrow{xy} \mathsf{t} \dashv A, y$ *and* $\mathsf{t}' = [f, [z/y]]\mathsf{t}$

Condition 1 ensures that communication with the environment occurs only along publicly known channels and that the knowledge of such channels is correctly propagated from one state to another when a transition occurs. Conditions 2 ensure that if a name can be received as input along a specific channel, then any other name can be received as well. Condition 3a asserts that transitions are preserved along injective renamings, while condition 3b shows how inputs of new names generate inputs of "old" names when moving from a name set to a larger one. Finally, condition 4 ensures that the transitions out of a state which has been injectively renamed are determined by those of the state itself.

In fact the definition contains some redundancy:

Proposition 3.4 *Condition 3b 'shifting' is implied by conditions 2a 'input-new' and 3a 'injective substitution'.*

Despite this we keep condition 3b, for two reasons. Firstly, we regard the condition as conceptually important, thus we did not want to omit it from the main definition. Secondly, conditions 2a,2b, introduced to ensure uniform behaviour of input transitions, can be argued to be unnecessary from the model-theoretic point of view (just as their analogues are neglected in the reduction of value-passing CCS to pure CCS [Mil89]). When 2a,2b are omitted, 3b becomes essential.

For illustrative purposes we list now a few simple consequences of Definition 3.3. Analogous properties of π-terms are often established as lemmas, e.g. to prove correspondence between labelled and reduction semantics (see [SV99a, Sew00] for explicitly-indexed developments).

Proposition 3.5 (Weakening) *If* $A \vdash \mathsf{s} \xrightarrow{\ell} \mathsf{t}$ *and* $x \notin A \cup \mathrm{fn}(\ell)$ *then* $\iota\mathsf{s} \xrightarrow{\ell} \jmath\mathsf{t}$, *where* $\iota : A \hookrightarrow A, x$ *and* $\jmath : A \cup \mathrm{fn}(\ell) \hookrightarrow (A \cup \mathrm{fn}(\ell)), x$.

Proposition 3.6 (Strengthening) *If* $A, x \vdash \iota\mathsf{s} \xrightarrow{\ell} \mathsf{t}'$, *and* $x \notin \mathrm{fn}(\ell)$, *where* $\iota : A \hookrightarrow A, x$, *then there exists* t *such that* $A \vdash \mathsf{s} \xrightarrow{\ell} \mathsf{t}$ *and* $\mathsf{t}' = \jmath\mathsf{t}$, *where* $\jmath : A \cup \mathrm{fn}(\ell) \hookrightarrow (A \cup \mathrm{fn}(\ell)), x$.

Proposition 3.7 (Converse of Shifting) *If* $A, y \vdash \iota\mathsf{s} \xrightarrow{xy} \mathsf{t}$, *where* $\iota : A \hookrightarrow A, y$, *then* $A \vdash \mathsf{s} \xrightarrow{xy} \mathsf{t}$.

Weakening and Strengthening are immediate consequences of conditions 3a and 4. The converse of Shifting requires 2b, strengthening and 3a.

4. Alternative indexing structure

There are several alternative indexing structures – one simpler, with only injective renaming; one more elaborate, with data for restriction; and variants of all with a chosen new-name function. We discuss the trade-offs briefly.

Sets and injections Instead of indexing by the category $\mathcal{N}$ one can index by $\mathcal{N}_{\mathrm{inj}}$, the subcategory of $\mathcal{N}$ with all objects but only injective functions as arrows. This gives a simpler structure, in which the transitions of a reindexed state $f\mathsf{s}$ are always determined by those of s. To make input prefix definable, however, the denotation of a process with n free names must be a function from n-tuples of names to $\mathcal{N}$-LTSs, not simply an $\mathcal{N}$-LTS – to define $[\![xy.P]\!]$ one would need $[\![\{^z/_y\}P]\!]$ for all z. Moreover, we doubt whether an analogue of the input axioms 2a,2b could be stated.

Building restriction into the indexing It is arguable that, as restriction is a fundamental π-calculus concept, one should take models with more data than our $\mathcal{N}$-LTSs, specifying how the transitions of states change when names are restricted. This leads to more complex axioms, though clearly also to a simpler definition of the restriction operator. In more detail, define $\mathcal{N}_\nu$ to be the category with objects finite subsets of $\mathcal{N}$ and arrows pairs $\langle f, R_f \rangle : A \to B$ where $f : A \rightharpoonup B$ is a partial function and $R_f \subseteq (A \setminus \mathrm{dom}(f)) \times (A \setminus \mathrm{dom}(f))$ is an equivalence relation. If $A \vdash \mathsf{s}$ then the re-indexing of s along $\langle f, R_f \rangle$ should be thought of as the state in which names in $A \setminus \mathrm{dom}(f)$ have been restricted, after being quotiented by R_f, and other names have been substituted as specified by f. Define composition of arrows by $\langle g, R_g \rangle \circ \langle f, R_f \rangle = \langle g \circ f, R_{g \circ f} \rangle$ where $R_{g \circ f} = R_f \cup \{ (a, a') \mid f(a)\ R_g\ f(a') \}$.

Choosing new names In our definition, for a state s above A, all names $w \notin A$ are treated symmetrically – corresponding to the operational fact that (if $x \in A$) there is a transition $A \vdash (\boldsymbol{\nu} z)\overline{x}z \xrightarrow{\overline{x}w} 0$ for *any* $w \notin A$. One can instead take a chosen new – a function $\boldsymbol{\nu} : \mathcal{P}_{\mathrm{fin}}(\mathcal{N}) \to \mathcal{N}$ such that $\forall A\ .\ \boldsymbol{\nu} A \notin A$. This leads to an endofunctor $\delta : \mathcal{N} \to \mathcal{N}$ defined by $\delta A = A \cup \{\boldsymbol{\nu} A\}$ and $\delta(f) = f \cup \{\boldsymbol{\nu} A \mapsto \boldsymbol{\nu} B\}$; the axioms can be restated in terms of δ. In this paper we have not taken a chosen new in order to keep the tight correspondence with the operational semantics, and for notational simplicity. The chosen new version of $\mathcal{N}_{\mathrm{inj}}$ is essentially the indexing structure used in [Sta96, FMS96, Hen96, CSW97].

5. Denotational semantics

We describe now operations on $\mathcal{N}$-LTS that we will use in giving a compositional semantics to the π-calculus. In [CS00] we turn the class of $\mathcal{N}$-LTS with initial name-set I into a category, $\mathcal{N}$-$\mathbf{LTS}_I$, in the obvious way. It is then straightforward to turn the operations below into functors, in fact into ω-continuous functors. The category $\mathcal{N}$-$\mathbf{LTS}_I$ can be easily shown to have colimits of ω-chains, in fact to be cocomplete (and complete). Thus a semantics of recursively defined processes, such as replicated ones, can be obtained using least fixed points of ω-chains in the usual way. Lacking enough space to develop all of the above (or a more concrete definition for replication), we have decided in this extended abstract not to consider recursive processes at all.

The most interesting operations are deadlock, which to obtain initiality has what may be a slightly surprising definition, and restriction and parallel composition. For restriction an equivalence relation, a semantic analogue of α-conversion, needs to be imposed on states – just as in the operational semantics a transition of $(\boldsymbol{\nu}x)P$ may be derived from a transition of $(\boldsymbol{\nu}\hat{x})\{^{\hat{x}}/_{x}\}P$ for any $\hat{x} \notin (\mathrm{fn}(P) \setminus x)$. For parallel, in the operational semantics states reachable by transitions from $P \mid Q$ may involve restriction of $P' \mid Q'$ for P', Q' reachable from P, Q. The construction over the model involves a similar quotienting as for restriction. The equivalence relation used in both cases is defined as follows.

Definition 5.1 *If $S : \mathcal{N} \to \mathbf{Set}$ is a functor and A is a finite subset of N, take $\leftrightarrow_A$ to be the equivalence relation on (possibly subsets of) the set $\coprod_{B \supseteq A} S(B)$ defined by $\langle B_1, s_1\rangle \leftrightarrow_A \langle B_2, s_2\rangle$ if there exists a bijection $b : B_1 \to_{\mathit{bij}} B_2$, such that for every $x \in A$, $b(x) = x$ and such that $S(b)(s_1) = s_2$.*

The equivalence classes of $\leftrightarrow_A$ are analogous to alpha-equivalence classes of terms w.r.t. renaming of names not in A. Observe that elements of $S(A)$ can only be related to themselves, i.e. their equivalence class is a singleton. For this reason, when no confusion arises, we will write s for $[\langle A, s\rangle]_{\leftrightarrow_A}$.

In the constructions below we shall often extend a transition system with new initial state over a chosen name set (say I), but now all of its reindexings must also be added. This can be expressed using the *representable* functor (see e.g. [MLM92]) $\mathcal{N}(I, -)$ which sends each name-set A to the set of functions (the morphisms in $\mathcal{N}$) from I to A. Given a function $g : A \to B$, $\mathcal{N}(I, g)(f : I \to A) = gf$. The new initial state is the identity on I, 1_I and each of its reindexings is given by the reindexing function itself. Notice that we write e.g. $S + \mathcal{N}(I, -)$ for the coproduct of functors which is given by the pointwise disjoint union of sets.

NOTATION: If U and V are two sets and no confusion arises, we will write $\mathsf{l} : U \to U \uplus V$ and $\mathsf{r} : V \to U \uplus V$ for the obvious left and right injections in their disjoint union. If $\mathsf{s} = \langle A, u\rangle$, write ls for $\langle A, \mathsf{l}u\rangle$ and similarly for r. In what follows, unless otherwise stated we suppose that $T = \langle S : \mathcal{N} \to \mathbf{Set}, \longrightarrow, \langle I, i\rangle\rangle$ and $T_k = \langle S_k : \mathcal{N} \to \mathbf{Set}, \longrightarrow_k, \langle I, i_k\rangle\rangle$ (for $k = 1, 2$) are $\mathcal{N}$-LTSs. Note that the initial name-sets I coincide.

Restriction If $T = \langle S : \mathcal{N} \to \mathbf{Set}, \longrightarrow, \langle (I, x), i\rangle\rangle$ define the restriction $\nu_{x \in (I,x)}(T)$ to be

$$\langle S' : \mathcal{N} \to \mathbf{Set}, \longrightarrow', \langle I, \mathsf{r}[\langle (I,x), i\rangle]_{\leftrightarrow_I}\rangle\rangle ,$$

where

- $S'(A) = S(A) \uplus (\coprod_{y \notin A} S(A, y))/{\leftrightarrow_A}$
- $\longrightarrow'$ is defined by the following three rules:

$$\frac{A \vdash \mathsf{s} \xrightarrow{\ell} \mathsf{t}}{A \vdash \mathsf{ls} \xrightarrow{\ell}{}' \mathsf{lt}}$$

$$\frac{A, z \vdash \mathsf{s} \xrightarrow{\ell} \mathsf{t}}{A \vdash \mathsf{r}[\mathsf{s}]_{\leftrightarrow_A} \xrightarrow{\ell}{}' \mathsf{r}[\mathsf{t}]_{\leftrightarrow_{A \cup \mathrm{fn}(\ell)}}}\ z \notin \mathrm{fn}(\ell)$$

$$\frac{A, z \vdash \mathsf{s} \xrightarrow{\overline{x}z} \mathsf{t}}{A \vdash \mathsf{r}[\mathsf{s}]_{\leftrightarrow_A} \xrightarrow{\overline{x}z}{}' \mathsf{l}[\mathsf{t}]_{\leftrightarrow_{A,z}}}\ x \neq z$$

Output and τ prefix If $x, y \in I$, define $\overline{x}y(T)$ to be

$$\langle S + \mathcal{N}(I, -) : \mathcal{N} \to \mathbf{Set}, \longrightarrow', \langle I, \mathsf{r}1_I\rangle\rangle ,$$

where $\longrightarrow'$ is defined by the following rules:

$$\frac{\mathsf{s} \xrightarrow{\ell} \mathsf{t}}{\mathsf{ls} \xrightarrow{\ell}{}' \mathsf{lt}} \qquad \frac{f : I \to A}{\langle A, \mathsf{r}f\rangle \xrightarrow{\overline{f(x)}f(y)} \langle A, \mathsf{l}S(f)(i)\rangle}$$

Define $\tau(T)$ similarly by labelling the transition in the first rule τ rather than $\overline{f(x)}f(y)$.

Input prefix If $T = \langle S : \mathcal{N} \to \mathbf{Set}, \longrightarrow, \langle (I, y), i\rangle\rangle$ is a transition system and $y \neq x \in I$, define $xy(T)$ to be

$$\langle S + \mathcal{N}(I, -) : \mathcal{N} \to \mathbf{Set}, \longrightarrow', \langle I, \mathsf{r}1_I\rangle\rangle ,$$

where

$$\frac{f : I \to A \qquad \iota : A \hookrightarrow A \cup \{z\}}{\langle A, \mathsf{r}f\rangle \xrightarrow{f(x)z}{}' \langle A \cup \{z\}, \mathsf{l}S([\iota f, [z/y]])(i)\rangle}$$

$$\frac{\mathsf{s} \xrightarrow{\ell} \mathsf{t}}{\mathsf{ls} \xrightarrow{\ell}{}' \mathsf{lt}}$$

Deadlock at I For every set of names I, define the deadlocked $\mathcal{N}$-LTS with free names in I as $0_I = \langle \mathcal{N}(I,-), \emptyset, \langle I, 1_I\rangle\rangle$. Notice that 0_I is the initial object of the category $\mathcal{N}$-$\mathbf{LTS}_I$ (see [CS00]).

Matching and Mismatching If $x, y \in I$, then define $[x = y](T)$ to be

$$\langle S + \mathcal{N}(I,-) : \mathcal{N} \to \mathbf{Set},\ \longrightarrow',\ \langle I, i\rangle\rangle\ ,$$

where $\longrightarrow'$ is defined by the following rules:

$$\frac{f : I \to A \qquad \langle A, S(f)i\rangle \xrightarrow{\ell} \langle B, s\rangle \qquad f(x) = f(y)}{\langle A, \mathsf{r}f\rangle \xrightarrow{\ell}{}' \langle B, \mathsf{l}s\rangle}$$

$$\frac{\mathsf{s} \xrightarrow{\ell} \mathsf{t}}{\mathsf{ls} \xrightarrow{\ell}{}' \mathsf{lt}}$$

Define $[x \neq y](T)$ similarly by requiring $f(x) \neq f(y)$ in the first rule.

Sum Define the sum,

$$T_1 + T_2 = \langle (S_1 + \mathcal{N}(I,-) + S_2) : \mathcal{N} \to \mathbf{Set},\ \longrightarrow_0,\ \langle I, \mathsf{m}1_I\rangle\rangle$$

where $\longrightarrow_0$ is defined by the following rules:

$$\frac{f : I \to A \quad \langle A, S_k(f)(i_1)\rangle \xrightarrow{\ell}_1 \mathsf{s}}{\langle A, \mathsf{m}f\rangle \xrightarrow{\ell}_0 \mathsf{ls}} \text{ (and sym. for } 2, \mathsf{r})$$

$$\frac{\mathsf{s} \xrightarrow{\ell}_1 \mathsf{t}}{\mathsf{ls} \xrightarrow{\ell}_0 \mathsf{lt}} \qquad\qquad \frac{\mathsf{s} \xrightarrow{\ell}_2 \mathsf{t}}{\mathsf{rs} \xrightarrow{\ell}_0 \mathsf{rt}}$$

where we now use $\mathsf{l}, \mathsf{m}, \mathsf{r}$ rather than just l and r.

If we assume the $\mathcal{N}$-LTSs to be non-restarting, we can give a more standard definition which (pointwise) glues together the two (sets of) initial states; in this case the sum is the categorical coproduct [CS00].

Parallel composition

NOTATION: If S_1 and S_2 are two functors $\mathcal{N} \to \mathbf{Set}$, and if $\leftrightarrow_A$ is the equivalence relation on $\coprod_{B \supseteq A}(S_1 \times S_2)(B) = \coprod_{B \supseteq A} S_1(B) \times S_2(B)$ defined as in Definition 5.1, and if $\mathsf{s}_1 = \langle B, s_1\rangle$ and $\mathsf{s}_2 = \langle B, s_2\rangle$ write $\mathsf{s}_1|_A\mathsf{s}_2$ for the equivalence class $[(B, s_1, s_2)]_{\leftrightarrow_A}$.

If T_1 and T_2 are two $\mathcal{N}$-LTSs as before, define their parallel composition,

$$T_1 | T_2 = \langle S_0 : \mathcal{N} \to \mathbf{Set},\ \longrightarrow_0,\ \langle I, \langle i_1, i_2\rangle\rangle\rangle$$

where

- $S_0(A) = (\coprod_{B \supseteq A}(S_1 \times S_2)(B))/\leftrightarrow_A$, while $S_0(f : A \to A')([(B, s_1, s_2)]_{\leftrightarrow_A}) = [(B', t_1, t_2)]_{\leftrightarrow_{A'}}$, where $t_k = S(f + g)(s_k)$, for $k = 1, 2$ and $g : B \setminus A \to_{\mathsf{bij}} B' \setminus A'$ is a bijection
- $\longrightarrow_0$ is defined by the following three rules (and symmetric versions of the first two):

$$\frac{A, A' \vdash \mathsf{s}_1 \xrightarrow{\ell}_1 \mathsf{t}_1 \qquad \iota : A, A' \hookrightarrow A \cup \mathrm{fn}(\ell), A'}{A \vdash \mathsf{s}_1|_A\mathsf{s}_2 \xrightarrow{\ell}_0 \mathsf{t}_1|_{A \cup \mathrm{fn}(\ell)}\iota\mathsf{s}_2}$$

$$\frac{A, A' \vdash \mathsf{s}_1 \xrightarrow{\overline{x}y}_1 \mathsf{t}_1 \qquad A, A' \vdash \mathsf{s}_2 \xrightarrow{xy}_2 \mathsf{t}_2}{A \vdash \mathsf{s}_1|_A\mathsf{s}_2 \xrightarrow{\tau}_0 \mathsf{t}_1|_A\mathsf{t}_2}$$

$$\frac{A, y \vdash \mathsf{s}_1|_{A,y}\mathsf{s}_2 \xrightarrow{\overline{x}y}_0 \mathsf{t}_1|_{A,y}\mathsf{t}_2}{A \vdash \mathsf{s}_1|_A\mathsf{s}_2 \xrightarrow{\overline{x}y}_0 \mathsf{t}_1|_{A,y}\mathsf{t}_2}$$

It is easy to verify that the functor S_0 is well defined, i.e. that the definition of $S_0(f)$ is independent of the choice of representatives and of the choice of the functions g.

Bisimilarity is defined in the usual way, but thanks to the indexing, we can also define directly in the model the closure under name substitutions, which for the π-calculus characterises the largest congruence included in bisimilarity.

Definition 5.2 *Define two $\mathcal{N}$-LTSs T_1 and T_2 to be strongly bisimilar if the LTS $\langle \mathsf{S}_1, \longrightarrow_1, \langle I, i_1\rangle\rangle$ and $\langle \mathsf{S}_2, \longrightarrow_2, \langle I, i_2\rangle\rangle$ are bisimilar in the usual sense of Milner [Mil89]. Say that they are strongly equivalent if, for every $f : I \to A$ the $\mathcal{N}$-LTSs $\langle S_1, \longrightarrow_1, \langle A, S_1(f)i_1\rangle\rangle$ and $\langle S_2, \longrightarrow_2, \langle A, S_2(f)i_2\rangle\rangle$ are strongly bisimilar.*

Weak bisimilarity and equivalence are defined similarly.

Exploiting the indexing structure even more, notice that if a bisimulation is further required to be a relation between S_1 and S_2 in the categorical sense, i.e. a subobject of the product $S_1 \times S_2$, one obtains an *open* bisimulation [San96b].

Compositional semantics to π-terms is given using the operations defined above in the obvious way. For a process term P, with free names in I, we write $[\![P]\!]_I$ for the corresponding $\mathcal{N}$-LTS. We conclude this section by stating the result which relates bisimulation in the model with early bisimulation in the operational semantics.

Theorem 5.3 *Let P and Q be two π-terms with free names in I. Then $P \dot{\sim}_I Q$ if and only if $[\![P]\!]_I$ is bisimilar to $[\![Q]\!]_I$.*

Proof:[Sketch] First of all observe that the operational semantics naturally induces for every process term P, with free names in I, an $\mathcal{N}$-LTS

$$(\![P]\!)_I = \langle \pi, \longrightarrow, \langle I, P\rangle\rangle\ ,$$

where $\pi(A) = \{Q \mid Q$ is a π-term and $\mathrm{fn}(Q) \subseteq A\}$, $\pi(f)$ simply relabels processes according to f, and there is a transition $\langle A, P\rangle \xrightarrow{\ell} \langle B, Q\rangle$ if $A \vdash P \xrightarrow{\ell} Q$ (according to the operational semantics) and $B = A \cup \mathrm{fn}(\ell)$. One can then prove by structural induction that $(\!|P|\!)_I$ is open bisimilar (cf. the remark after Definition 5.2) to $[\![P]\!]_I$. The theorem is now an easy consequence of this last statement.

6. $\mathcal{N}$-LATS

In this section we define a class of causal models by smoothly lifting the notion of labelled asynchronous[1] transition system [Bed88, Shi85, WN95] (LATS for short) to our indexed setting. LATS are a simple extension of standard LTS in which transitions have both standard labels and *events*, upon which an independence relation is defined. Roughly speaking, concurrency is modelled by requiring that transitions tagged with independent events might occur in any order. As discussed in the introduction, in π-calculi dependencies between transitions may arise from their name usage:

Definition 6.1 *If A is a set of names and ℓ_1 and ℓ_2 are two labels, we say that ℓ_2 is A-dependent on ℓ_1 if one of the following two cases applies:*

1. $\mathrm{val}(\ell_1) = \mathrm{chan}(\ell_2) \not\subseteq A$
2. $\mathrm{val}(\ell_1) = \mathrm{val}(\ell_2) \not\subseteq A$, *one of ℓ_1, ℓ_2 is an input action and the other is an output action.*

Definition 6.2 *Define an Indexed LATS ($\mathcal{N}$-LATS) to be a structure*

$$T = \langle S : \mathcal{N} \to \mathbf{Set},\ \longrightarrow,\ \langle I, i\rangle,\ E,\ \mathcal{I}\rangle$$

where $\langle I, i\rangle \in \mathsf{S}$,

$$\longrightarrow\ \subseteq \mathsf{S} \times (\mathrm{Lab} \times E) \times \mathsf{S}\ ,$$

E is a set of events, $\mathcal{I} \subseteq E \times E$ is an independence relation between events and the following conditions hold.

1. *For every event $e \in E$, the structure $\langle S : \mathcal{N} \to \mathbf{Set},\ \underset{e}{\longrightarrow},\ \langle I, i\rangle\rangle$ is a transition system according to Definition 3.3, where $\underset{e}{\longrightarrow}$ is the set $\{\,\langle \mathsf{s}, \ell, \mathsf{t}\rangle \mid \langle \mathsf{s}, \ell, e, \mathsf{t}\rangle \in\longrightarrow\,\}$.*
2. *$\mathcal{I}$ is irreflexive and symmetric*
3. *If $A \vdash \mathsf{s} \xrightarrow[e_1]{\ell_1} \mathsf{t}$, and $\mathsf{t} \xrightarrow[e_2]{\ell_2} \mathsf{u}$, and $e_1 \mathcal{I} e_2$, and moreover ℓ_2 is not A-dependent on ℓ_1, then there exists a state t' such that $\mathsf{s} \xrightarrow[e_2]{\ell_2} \mathsf{t}'$ and $\mathsf{t}' \xrightarrow[e_1]{\ell_1} \mathsf{u}$.*

[1] There is an unfortunate clash of terminology here: this usage of 'asynchronous' is unrelated to the usage describing process calculi without output prefixing.

Often LATS are defined using more axioms (see [Bed88, WN95]). Here we have decided to keep the axiomatisation as light as possible, as none of the extra axioms is directly relevant for the definability of the semantic constructions that we consider. Moreover we allow the same event to carry different labels. This is particularly useful in coping with the proliferation of transitions induced by reindexing and by the input actions. It is not difficult to devise simple variations of our definition which adhere more closely to the traditional case.

Building on the independence relation, transitions occurring in a run of a process can be given a causal partial order describing which transitions are necessary conditions for the occurrence of others. Roughly speaking one transition causes the following one if the corresponding events are not independent of each other. As discussed in the introduction, one can choose whether or not to consider name dependencies – for $\mathcal{N}$-LATS there are two natural ways of defining partial orders out of runs, one taking account only of the independence relation and another which also takes name dependencies into account.

NOTATION: For every natural number n, write $[n]$ for the set $\{k \mid 1 \le k \le n\}$. Observe that, in particular, $[0] = \emptyset$.

Definition 6.3 *For every run r*

$$A_0 \vdash \mathsf{s}_0 \xrightarrow[e_1]{\ell_1} \mathsf{s}_1 \xrightarrow[e_2]{\ell_2} \mathsf{s}_2 \cdots \xrightarrow[e_n]{\ell_n} \mathsf{s}_n$$

of an $\mathcal{N}$-LATS we define two labelled partial orders:

1. *Define $\mathrm{po}(r)_{\mathcal{I}} = \langle [n], \trianglelefteq^r_{\mathcal{I}}, l^r\rangle$, where*
   - *(a) n is the length of the run r.*
   - *(b) $\trianglelefteq^r_{\mathcal{I}}$ is the transitive closure of $\preceq^r_{\mathcal{I}}$ which is defined as $i \preceq^r_{\mathcal{I}} j$ if $i \le j$ and $\neg(e_i \mathcal{I} e_j)$*
   - *(c) $l^r(k) = \ell_k$, for every $k \in [n]$*
2. *Define $\mathrm{po}(r)_{\mathcal{I}D} = \langle [n], \trianglelefteq^r_{\mathcal{I}D}, l^r\rangle$, where n and l^r are obtained as above, while $\trianglelefteq^r_{\mathcal{I}D}$ is the transitive closure of $\preceq^r_{\mathcal{I}D}$ which is defined as $i \preceq^r_{\mathcal{I}D} j$ if $i \le j$ and either $\neg(e_i \mathcal{I} e_j)$ or ℓ_j is A_i-dependent on ℓ_i, where $\mathsf{s}_i = (A_i, s_i)$.*

History preserving bisimulation [RT88, GG89, DDNM88b] is a bisimulation between runs of processes which accounts for causality by requiring related runs to originate isomorphic partial orders of transitions:

Definition 6.4 *Let T_1 and T_2 be two $\mathcal{N}$-LATSs with initial name-set I and let $\mathrm{Run}(T_i)$ (for $i = 1, 2$) be the corresponding sets of runs. A relation $\mathcal{B} \subseteq \mathrm{Run}(T_1) \times \mathrm{Run}(T_2)$ is an history preserving bisimulation (hpb) if it satisfies the following conditions*

1. $\langle I \vdash \langle I, i_1\rangle, I \vdash \langle I, i_2\rangle\rangle \in \mathcal{B}$

2. $\langle r_1, r_2\rangle \in \mathcal{B}$ *implies*

   (a) $\mathrm{po}(r_1)_{\mathcal{I}} = \mathrm{po}(r_2)_{\mathcal{I}}$

   (b) *if* r_1' *extends* r_1 *with a transition* $\mathsf{s}_n \xrightarrow[e_{n+1}]{\ell_{n+1}}_1 \mathsf{s}_{n+1}$ *then there exists a run* r_2' *which extends* r_2 *with a transition* $\bar{\mathsf{s}}_n \xrightarrow[\bar{e}_{n+1}]{\ell_{n+1}}_2 \bar{\mathsf{s}}_{n+1}$ *such that* $\langle r_1', r_2'\rangle \in \mathcal{B}$

   (c) *the symmetric condition to the above.*

The relation $\mathcal{B}$ is a name-dependency aware hpb (ndahpb) if the condition $2(a)$ *is changed into* $\mathrm{po}(r_1)_{\mathcal{I}D} = \mathrm{po}(r_2)_{\mathcal{I}D}$.

The constructions of Section 5 can be easily adapted to become constructions on $\mathcal{N}$-LATS. We shall now briefly indicate how they need to be extended to take account of the presence of events and of the independence relation. In all rules where a label is carried from the premise to the conclusion, the event is also carried (suitably injected).

Restriction The set of events and the independence relation does not change.

Prefixes A new event, not in the independence relation with any other is added and it decorates all of the new transitions.

Deadlock The set of events is empty and so is the independence relation.

Matching and Mismatching Events and the independency relation are left untouched.

Sum The set of events is taken to be the disjoint union of the originals but no new independence pairs are added.

Parallel composition If E_1 and E_2 are the two sets of events we define E_0 to be the disjoint union $E_1 \uplus (E_1 \times E_2) \uplus E_2$. Writing this as $(E_1 \times \{\star\}) \cup (E_1 \times E_2) \cup (\{\star\} \times E_2)$ for $\star \notin E_1 \cup E_2$, the independence relation is defined by $\langle e_1, e_2\rangle \mathcal{I}_0 \langle e_1' e_2'\rangle$ if both $e_1 \hat{\mathcal{I}}_1 e_1'$ and $e_2 \hat{\mathcal{I}}_2 e_2'$, where $\hat{\mathcal{I}}_k$ is the union of $\mathcal{I}_k$ and $\langle \star, \star\rangle$. The new τ-transitions are decorated by the pairs of enabling events.

Process terms can then be given a denotational semantics and then related by (nda) history preserving bisimilarity. In the remainder of this section we will mostly concentrate on the relationship between our semantics and the causal bisimulation of [BS98]. In particular we present correspondence results relating our hbp semantics to causal bisimulation, and further discuss name-dependency.

In the paper [BS98], no notion of strong bisimulation is defined. The authors in fact defined directly causal bisimulation in the weak, i.e. abstracting away from τ actions, form. To match with our definitions we therefore need either to define weak history-preserving bisimulation or to modify their setting in order to make τ actions, and not their effect only, visible. We will in fact do both, ending up with two correspondence results, one for strong and one for weak bisimulation. Due to space constraints we cannot report the definition of causal bisimulation here and therefore refer to *loc. cit.* for definitions and discussions of the relevance of their approach. We simply mention here, using their notation, what modifications are needed in order to define strong causal bisimulation.

Definition 6.5 *An operational semantics for* strong causal bisimulation *is obtained by modifying the definition of [BS98, table 3, page 365] in the rules which derive silent actions in the following way:*

T-PRE, T-SUM, T-PAR, T-RES, T-CAU, T-REP*: These are all subsumed in the homologous rules,* OUT, SUM, PAR, RES, CAU, REP, *respectively, which were originally defined for non-τ actions.*

COM*: This is changed as follows (recall that we are dealing with the monadic π-calculus):*

$$\frac{A_1 \xrightarrow[K_1:k]{(\boldsymbol{\nu} z)\overline{x}y} A_1' \qquad A_2 \xrightarrow[K_2:k]{xy} A_2'}{A_1 \mid A_2 \xrightarrow[K_1 \cup K_2:k]{\tau} (\boldsymbol{\nu} z)(A_1'[k \rightsquigarrow (K_2, k)] \mid A_2'[k \rightsquigarrow (K_1, k)])}$$

with conditions $z \notin \mathrm{fn}(A_2)$ and $k \notin \mathcal{K}(A_1, A_2)$.[2]

Strong causal bisimulation can now be defined in the usual way, by requiring transitions to agree not only on the labels but on the causes too. We can now state our first non-interleaving correspondence result:

Theorem 6.6 *Let P and Q be two terms of the π-calculus with free names in I and let $[\![P]\!]_I^c$ and $[\![Q]\!]_I^c$ be their interpretations as $\mathcal{N}$-LATS's. Then $[\![P]\!]_I^c$ is history preserving bisimilar to $[\![Q]\!]_I^c$ if and only if P is strongly causal bisimilar to Q.*

A weak version of history preserving bisimulation can be given in the spirit of [Vog95].

Definition 6.7 *Let r be a run in an asynchronous transition system, let n be the length of r and let n^τ be the number of transitions in r which are not labelled τ. For every $i \leq n^\tau$, define $n_i \leq n$ inductively as follows: n_1 is the smallest number h such that the h-th transition of r has label $\ell_h \neq \tau$; n_{j+1} is the smallest number h such that the h-th transition of r has label $\ell_h \neq \tau$ and that moreover is strictly bigger than n_j.*

Starting with a run r of an $\mathcal{N}$-LATS, by means of the above definition, we can define partial orders of observable events in runs as follows:

[2] In [BS98] the notation $(\boldsymbol{\nu} z)\overline{x}y$ is employed for possibly-bound outputs.

Definition 6.8 *Let r be a run of an $\mathcal{N}$-LATS and let* $\mathrm{po}(r)_{\mathcal{I}}$ *and* $\mathrm{po}(r)_{\mathcal{I}D}$ *be the corresponding partial orders as in Definition 6.3. Define* $\mathrm{po}(r)_{w\mathcal{I}}$ *and* $\mathrm{po}(r)_{w\mathcal{I}D}$ *to be the partial orders* $\langle [n^r], \trianglelefteq_{w\mathcal{I}}, l^r_w \rangle$ *and* $\langle [n^r], \trianglelefteq_{w\mathcal{I}D}, l^r_w \rangle$*, respectively, where* $l^r_w(i) = l^r(n_i)$*,* $i \trianglelefteq_{w\mathcal{I}} j$ *if* $n_i \trianglelefteq_{\mathcal{I}} n_j$ *and* $i \trianglelefteq_{w\mathcal{I}D} j$ *if* $n_i \trianglelefteq_{\mathcal{I}D} n_j$.

Weak history preserving bisimulations are now defined as relations between runs as in Definition 6.4 but where, as usual, "strong" transitions $\mathsf{s}_1 \xrightarrow[e_1]{\ell}_1 \mathsf{t}_1$ are simulated by "weak" ones $\mathsf{s}_2 \overset{\hat{\ell}}{\underset{e_2}{\Longrightarrow}}_2 \mathsf{t}_2$ (and symmetrically) and with condition $2(a)$ replaced by $\mathrm{po}(r)_{w\mathcal{I}} = \mathrm{po}(r')_{w\mathcal{I}}$ or by $\mathrm{po}(r)_{w\mathcal{I}D} = \mathrm{po}(r')_{w\mathcal{I}D}$, for the name-dependency aware case. We can then prove the following result:

Theorem 6.9 *Let P and Q be two terms of the π-calculus with free names in I and let $[\![P]\!]^c_I$ and $[\![Q]\!]^c_I$ be their interpretations as $\mathcal{N}$-LATSs. Then $[\![P]\!]^c_I$ is weak history preserving bisimilar to $[\![Q]\!]^c_I$ if and only if P is causal bisimilar to Q in the sense of [BS98].*

In [BS98] it is argued that, because of the dependencies due to the binding of names, processes like $(\boldsymbol{\nu} y)(\overline{x}y.\overline{y}z)$ and $(\boldsymbol{\nu} y)(\overline{x}y|\overline{y}z)$ should be indistinguishable by an external observer. Nonetheless causal bisimulation distinguishes them, as it only tracks the dependencies due to the structure of processes – in the example, one output is prefixing the other in the first process but not in the second. The paper leaves open the possibility of a further refinement of the treatment of causes in the operational semantics to identify the above two processes.

Their remark has been tackled in [JJ95], where a domain model of π-terms based on Kahn networks is presented. There the induced equivalence equates the two processes, but it seems to us that the equivalence is anyway a traced-based rather than a bisimulation based one. In [DP99], the authors use the combination of different partial orders to achieve the effect of equating the two processes above. In this paper we instead refined the way the causal order of events in a run is determined. This has led to the notion of name-dependency aware history preserving bisimulation defined above. It is easy to verify that name-dependency aware history preserving bisimilarity is a coarser relation than history preserving bisimilarity and that the former equates the two example processes:

Proposition 6.10 *If two asynchronous transition systems are history preserving bisimilar than they are name-dependency aware history preserving bisimilar.*

Proposition 6.11 *The denotations of the process terms $(\boldsymbol{\nu} y)(\overline{x}y.\overline{y}z)$ and $(\boldsymbol{\nu} y)(\overline{x}y|\overline{y}z)$ are name-dependency aware history preserving bisimilar but not history preserving bisimilar.*

References

[BC88] G. Boudol and I. Castellani. Permutation of transitions: an event structure semantics for CCS and SCCS. In *Proceedings of REX School/Workshop*, volume 354 of *LNCS*, pages 411–427, 1988.

[BDN95] M. Boreale and R. De Nicola. Testing equivalences for mobile processes. *Information and Computation*, 120:279–303, 1995.

[Bed88] M. Bednarczyk. *Categories of Asynchronous Systems*. PhD thesis, University of Sussex, 1988.

[BG95] N. Busi and R. Gorrieri. A petri net semantics for π-calculus. In *Proceedings of CONCUR'95*, volume 962 of *LNCS*, pages 145–159, 1995.

[Bou92] G. Boudol. Asynchrony and the π-calculus. Technical Report 1702, INRIA, Sophia Antipolis, 1992.

[BS98] M. Boreale and D. Sangiorgi. A fully abstract semantics for causality in the π-calculus. *Acta Informatica*, 35:353–400, 1998.

[Cat99] G. L. Cattani. *Presheaf Models for Concurrency*. PhD thesis, University of Aarhus, 1999.

[CS00] G. L. Cattani and P. Sewell. Models for name-passing processes: Interleaving and causal. Technical report, Cambridge University Computer Laboratory, 2000.

[CSW97] G. L. Cattani, I. Stark, and G. Winskel. Presheaf models for the π-calculus. In *Proceedings of CTCS'97*, volume 1290 of *LNCS*, pages 106–126, 1997.

[CW97] G. L. Cattani and G. Winskel. Presheaf models for concurrency. In *Proceedings of CSL'96*, volume 1258 of *LNCS*, pages 58–75, 1997.

[CW99] G. L. Cattani and G. Winskel. Presheaf models for CCS-like languages. Technical Report 477, Cambridge University Computer Laboratory, 1999. Submitted for publication.

[DD89] Ph. Darondeau and P. Degano. Causal trees. In *Proceedings of ICALP'89*, volume 372 of *LNCS*, pages 234–248, 1989.

[DDNM88a] P. Degano, R. De Nicola, and U. Montanari. On the consistency of "truly concurrent" operational and denotational semantics (extended abstract). In *Proceedings of LICS'88*, pages 133–141. IEEE, 1988.

[DDNM88b] P. Degano, R. De Nicola, and U. Montanari. Partial orderings descriptions and observations of nondeterministic concurrent processes. In *Proceedings of REX School/Workshop*, volume 354 of *LNCS*, pages 438–496, 1988.

[DP99] P. Degano and C. Priami. Non-interleaving semantics for mobile processes. *Theoretical Computer Science*, 216(1-2):237–270, 1999.

[FCW99] M. P. Fiore, G. L. Cattani, and G. Winskel. Weak bisimulation and open maps (extended abstract). In *Proceedings of LICS'99*, pages 67–76. IEEE, 1999.

[FMS96] M. P. Fiore, E. Moggi, and D. Sangiorgi. A fully-abstract model for the π-calculus (extended abstract). In *Proceedings of LICS'96*, pages 43–54. IEEE, 1996.

[FPT99] M. P. Fiore, G. D. Plotkin, and D. Turi. Abstract syntax and variable binding. In *Proceedings of LICS'99*, pages 193–202. IEEE, 1999.

[GG89] R. van Glabbeek and U. Goltz. Equivalence notions for concurrent systems and refinement of actions. In *Proceedings of MFCS'89*, volume 379 of *LNCS*, pages 237–248, 1989.

[GP99] M. Gabbay and A. Pitts. A new approach to abstract syntax involving binders. In *Proceedings of LICS'99*, pages 214–224. IEEE, 1999.

[Hen96] M. Hennessy. A fully abstract denotational semantics for the π-calculus. Technical Report 96:04, School of Cognitive and Computing Sciences, University of Sussex, 1996. To appear in *Theoretical Computer Science*.

[Hof99] M. Hofmann. Semantical analysis of higher-order abstract syntax. In *Proceedings of LICS'99*, pages 204–213. IEEE, 1999.

[Hon99] K. Honda. Behavioural subtyping in name passing synchronisation trees. Available at `http://www.dcs.qmw.ac.uk/~kohei/`, 1999.

[HT91] K. Honda and M. Tokoro. An object calculus for asynchronous communication. In *Proceedings of ECOOP'91*, volume 512 of *LNCS*, pages 133–147, 1991.

[JJ95] L. J. Jagadeesan and R. Jagadeesan. Causality and true concurrency: A data-flow analysis of the pi-calculus (extended abstract). In *Proceedings of AMAST'95*, volume 936 of *LNCS*, pages 277–291, 1995.

[JNW96] A. Joyal, M. Nielsen, and G. Winskel. Bisimulation from open maps. *Information and Computation*, 127(2):164–185, 1996.

[Kie94] A. Kiehn. Comparing locality and causality based equivalences. *Acta Informatica*, 31:697–718, 1994.

[Mil89] R. Milner. *Communication and Concurrency*. Prentice Hall, 1989.

[MLM92] S. Mac Lane and I. Moerdijk. *Sheaves in Geometry and Logic: A First Introduction to Topos Theory*. Springer-Verlag, 1992.

[MP95] U. Montanari and M. Pistore. Concurrent semantics for the π-calculus. In *Proceedings of MFPS XI*, volume 1 of *ENTCS*, pages 337–356. Elsevier, 1995.

[MP98] U. Montanari and M. Pistore. An introduction to history dependent automata. In *Second Workshop on Higher-Order Operational Techniques in Semantics (HOOTS II)*, volume 10 of *ENTCS*. Elsevier, 1998.

[MPW92] R. Milner, J. Parrow, and D. Walker. A calculus of mobile processes. Part I and II. *Information and Computation*, 100(1):1–77, 1992.

[MPW93] R. Milner, J. Parrow, and D. Walker. Modal logics for mobile processes. *Theoretical Computer Science*, 114(1):149–171, 1993.

[Pau98] L. Paulson. The inductive approach to verifying cryptographic protocols. *Journal of Computer Security*, 6:85–128, 1998.

[RT88] A. Rabinovitch and B. Traktenbrot. Behaviour structures and nets. *Fundamenta Informatica*, 11(4):357–404, 1988.

[San93] D. Sangiorgi. *Expressing Mobility in Process Algebras: First-Order and Higher-Order Paradigms*. PhD thesis, University of Edinburgh, 1993.

[San96a] D. Sangiorgi. π-calculus, internal mobility, and agent-passing calculi. *Theoretical Computer Science*, 167(2):235–274, 1996.

[San96b] D. Sangiorgi. A theory of bisimulation for the π-calculus. *Acta Informatica*, 33(1):69–97, 1996.

[Sel97] P. Selinger. First order axioms for asynchrony. In *Proceedings of CONCUR'97*, volume 1243 of *LNCS*, pages 376–390, 1997.

[Sew97] P. Sewell. On implementations and semantics of a concurrent programming language. In *Proceedings of CONCUR'97*, volume 1243 of *LNCS*, pages 391–405, 1997.

[Sew00] P. Sewell. Pi calculi. In H. Bowman and J. Derrick, editors, *Formal Methods for Distributed Processing, An Object Oriented Approach*. CUP, 2000. To appear.

[Shi85] M. W. Shields. Concurrent machines. *Theoretical Computer Science*, 28:449–465, 1985.

[SNW96] V. Sassone, M. Nielsen, and G. Winskel. Models for concurrency: towards a classification. *Theoretical Computer Science*, 170(1-2):297–348, 1996.

[Sta96] I. Stark. A fully abstract domain model for the π-calculus. In *Proceedings of LICS'96*, pages 36–42. IEEE, 1996.

[SV99a] P. Sewell and J. Vitek. Secure composition of insecure components. In *Proceedings of the 12th IEEE Computer Security Foundations Workshop*, pages 136–150. IEEE, 1999. Extended version as University of Cambridge TR 463, 1999.

[SV99b] P. Sewell and J. Vitek. Secure composition of untrusted code: Wrappers and causality types. Technical Report 478, Computer Laboratory, University of Cambridge, 1999. To appear in *Proceedings of the 13th IEEE Computer Security Foundations Workshop*.

[Vog95] W. Vogler. Generalized OM-bisimulation. *Information and Computation*, 118(1):38–47, 1995.

[WN95] G. Winskel and M. Nielsen. Models for concurrency. In *Handbook of logic in computer science, Vol. 4*, Oxford Sci. Publ., pages 1–148. OUP, 1995.