Locality and Non-interleaving Semantics in calculi for mobile processes*

Davide Sangiorgi†

January 29, 1994

Abstract

Process algebra semantics can be categorised into *non-interleaving* semantics, where parallel composition is considered a primitive operator, and *interleaving* semantics, where concurrency is reduced to sequentiality plus nondeterminism. The former have an appealing intuitive justification, but the latter are mathematically more tractable.

This paper addresses the study of non-interleaving semantics in the framework of process algebras for mobile systems, like π-calculus [MPW92, Mil91]. We focus on *location bisimulation* ($\approx_\ell$), in our opinion one of the most convincing non-interleaving equivalences, which aims to describe the *spatial* dependencies on processes. We introduce $\approx_\ell$ in π-calculus following the definition for CCS given in [BCHK91b]. Our main contribution is to show that in π-calculus $\approx_\ell$ can be expressed, or implemented, within the ordinary interleaving *observation equivalence* [Mil89, MPW92] by means of a fairly simple and fully abstract encoding. Thus, we can take advantage of the easier theory of observation equivalence to reason about $\approx_\ell$. We illustrate this with a few examples, including the proof of the congruence properties of $\approx_\ell$. We show that in π-calculus $\approx_\ell$ is not a congruence, and that the full abstraction of the encoding extends to the induced congruence.

The results in the paper also shed more light on the expressive power of the π-calculus.

*An extract from this paper has appeared in the proc. International Conference on Theoretical Aspects of Computer Software (TACS'94), Japan, 1994.

†Address: Laboratory for Foundations of Computer Science, Department of Computer Science, University of Edinburgh, The King's Buildings, Edinburgh EH9 3JZ, U.K.
Email: `sad@dcs.ed.ac.uk`

1 Introduction

Process algebras are one of the most successful formalism for the description and analysis of concurrent systems. Roughly, process algebra semantics can be categorised into *non-interleaving* (sometimes also called *true-concurrency*) semantics, where parallel composition is considered a primitive operator, and *interleaving* semantics, where concurrency is reduced to sequentiality plus nondeterminism. The classical example which distinguishes between them is the equality between $a|b$ and $a.b+b.a$, which only holds in the interleaving approach.

Non-interleaving semantics have an appealing intuitive justification. They can capture — to different extents — the concurrency and the independence among the activities of a system. By contrast, interleaving semantics have the advantage of an easier mathematical treatment which has contributed to the development of better tools for the investigation of the properties of the resulting behavioural equivalences. The superior tractability is evident in the axiomatisations of the finite processes. For an interleaving semantics the argument is often pretty much standard, with the sum operator playing a central role in the definition of the normal forms. A non-interleaving equivalence usually requires more complicated rules and completeness proof, besides the introduction of extra operators, i.e. operators extraneous to the basic language. (Indeed, extra operators are often used even in the definition itself of the equivalence, an example being the location prefix of location bisimulation discussed below). Among the interleaving equivalences, *observation equivalence* is one of the most studied; a detailed exposition of its congruence properties, axiomatisation, and logic characterisation in CCS can be found in Milner's book [Mil89].

This paper addresses the study of non-interleaving semantics in the framework of process algebras for mobile systems. These are network of processes whose communication structure may vary as the computation proceeds. The best example of such process algebras is π-calculus [MPW92], an extension of CCS [Mil89] proposed by Milner, Parrow and Walker in *op. cit.* and later refined by Milner [Mil91]. In π-calculus, processes interact with each other by exchanging *channels*, also called *names* or *ports*. The most influential feature of π-calculus w.r.t. its predecessor CCS is the treatment of the operator of name restriction, whose scope may be dynamically modified as effect of process interactions. This, for instance, allows us to effectively model the creation of *unique* names (i.e. names different from all others) and the passing of references, pointers and resource ownerships.

Up-to-now an intensive effort has been devoted to the study of the interleaving semantics for π-calculus, but nothing has yet been done on the non-interleaving side. Our claim — and the main motivation for the work developed in this paper — is that π-calculus is a language very well-suited to the study of true-concurrency. Various behavioural equivalences which belong to this area have been formulated using special machineries of reference- or pointer-like objects; examples are the *location bisimulation* of [BCHK91b], the *causal bisimulation* of [DD89] or the *ST-split semantics* of [GV87, GL91]. These instruments are intrinsic ingredients of π-calculus and it is reasonable to ask ourselves whether this fact can be exploited in any significant way.

In the non-interleaving spectrum, we can distinguish two important subgroups, aiming, respectively, at the description of the causal and of the spatial dependencies in processes. The two approaches yield semantics in general different, but both are important for they support different views of a concurrent system. In this paper we focus on the second subgroup; more precisely, on *location bisimulation* ($\approx_\ell$), in our opinion the most convincing spatial-sensitive equivalence. An extended motivation for $\approx_\ell$ is presented in [BCHK91a], and includes the case of local deadlock, i.e. deadlock in one of the parallel components of a system, and the case in which the assignment of processes to processors has to be taken into account.

The idea of location bisimulation is that by interacting with the system, an observer also detects the location where the action comes from. Thus, $P \mid Q$ is viewed as a distributed system where the subprocesses P and Q act at differentiated sites. Locations are assigned dynamically, at the act of action-observation. This requires, on the one hand, developing an operational semantics for the language which replaces statements of the form $P \xrightarrow{\mu} P'$ with statements of the form $P \xrightarrow[u]{\mu} P'$; the word u represents the location of the action and is constructed from a distinguished alphabet of atomic locations. On the other hand, it requires extending the syntax of the language with a *location prefix* $u :: P$, to record the assignment of locations to processes. The relation $\approx_\ell$ is defined on top of these enriched language and transition system using familiar bisimulation techniques.

We introduce location bisimulation in π-calculus following the definition in CCS given in [BCHK91b]. For this, π-calculus's standard syntax and transition system have to be enriched with location names. Our main contribution is to show that the location machinery of $\approx_\ell$ can be *implemented* in π-calculus, so that the study of $\approx_\ell$ can be carried out from within the ordinary interleaving observation equivalence ($\approx$). The encoding $\mathcal{S}$ which realises this acts as a homomorphism on all operators but prefixes, where a component is added to access the location informations. There is a close operational correspondence between a process P and its encoding $\mathcal{S}[\![P]\!]$. Moreover, the encoding exactly characterises location bisimulation, i.e. for each name y (the encoding is parametric on a name)

$$P \approx_\ell Q \quad \text{ iff } \quad \mathcal{S}[\![P]\!]\langle y \rangle \approx \mathcal{S}[\![Q]\!]\langle y \rangle \, .$$

The full abstraction allows us to take advantage of the simpler theory of $\approx$ to reason about $\approx_\ell$. We illustrate this with a few examples, including the proof of the congruence properties of $\approx_\ell$. We shall see that $\approx_\ell$, in the same way as $\approx$, is not a congruence, and that the full abstraction of $\mathcal{S}$ extends to the induced congruences, respectively called *location congruence* and *observation congruence*.

The possibility of reconciliating interleaving and non-interleaving semantics already emerges, for instance, in the works of Degano, De Nicola and Montanari [DDM87, DDM93], who have shown that, in CCS, the differences between the two approaches can be imputed to the assumptions made on the observability of systems. The results in this paper represent a further contribution in the same direction. They also shed more light on the expressive power of π-calculus, and accompany, for instance, the results in [Mil92, San92] on the encoding of λ-calculus and of higher-order calculi, and in [Wal93] on the encoding of object-oriented languages. It is worth mentioning the proof techniques we employ: Some of these are based on recently developed theories, such as *expansions* [AKH92, SM92] and *bisimulations up-to context* [San94]. Hence the proofs provide us with examples where these techniques are necessary or very helpful.

References for location bisimulation. Location bisimulation has been introduced by Boudol *et al.* in [BCHK91a] and then rectified in [BCHK91b]. It (or variants of it) has been studied by Aceto [Ace91], Mukund and Nielsen [MN92], Castellani [Cas93] and Murphy [Mur93], who have explored the possibility of assigning locations to actions statically, rather than dynamically at the moment of inferring the action; and by Kiehn [Kie91], Montanari and Yankelevich [MY92], Degano and Priami [DP92], Corradini and De Nicola [CD93] who were interested in developing general or parametric frameworks in which to compare spatial- and causal-sensitive equivalences.

Structure of the paper. In Section 2 we review π-calculus's standard syntax and observation equivalence. Section 3 presents the expansion relation and a few "up-to" techniques, which will be utilised in various proofs in the paper. In Section 4 we introduce location bisimulation in π-calculus following what Boudol *et al.* have done for CCS in [BCHK91b]. In Section 5 we present the encoding $\mathcal{S}$, together with a few examples to show how it works. To analyse $\mathcal{S}$, it is convenient to extend it to an encoding $\mathcal{L}$ of located processes, i.e. processes which may also contain locating prefixes.

Sections 6-9 are devoted to the proof of the correctness of $\mathcal{L}$ — which implies the correctness of $\mathcal{S}$. Sections 7 and 9 constitute the core of the proof: In the former we establish the operational correspondence yielded by $\mathcal{L}$, both on strong and on weak transitions; in the latter its full abstraction. In Sections 6 and 8 we demonstrate a few auxiliary results about the process components which implement the location machinery of location bisimulation. In particular, we show how to eliminate these components when they appear at the outermost level of processes, i.e. non underneath a prefix. This is very useful to facilitate the reasoning about processes returned by the encoding.

In Section 10 we apply the full abstraction of $\mathcal{S}$ and $\mathcal{L}$ to derive properties of location bisimulation and congruence. Finally, in Section 11 we report some conclusions and possible future work.

2 The polyadic π-calculus

We start by reviewing the standard syntax and the interleaving semantics of the polyadic π-calculus, as in [Mil91]. Later, in Section 4, we shall enrich the syntax of processes with a location prefix "$u :: -$" and on the resulting *located processes* we shall define the non-interleaving *location bisimulation.* There are transition and algebraic rules (namely, those of Tables 1 and 2) which we want to use both on the standard processes and on the located processes. To avoid repetitions, we write these rules using special *metavariables* $X, Y, \ldots$ which, according to the case, are supposed to be instantiated with standard or located processes. We write the composition of two relations $\mathcal{R}$ and $\mathcal{R}'$ as $\mathcal{R}\,\mathcal{R}'$.

2.1 Syntax of standard processes

Letters $a, b, \ldots, x, y, \ldots$ range over the infinite set of names, and D over the set of constant symbols. The class $\mathcal{P} = \{P, Q, R, \ldots\}$ of the *standard π-calculus processes* is built from the operators of inaction, input and output prefix, sum, parallel composition, restriction, matching and constant application:

$$P \; := \; \mathbf{0} \; \mid \; a(\tilde{b}).\, P \; \mid \; \overline{a}\langle\tilde{b}\rangle.\, P \; \mid \; P_1 + P_2 \; \mid \; P_1 \mid P_2 \; \mid \; \boldsymbol{\nu}\, a\, P \; \mid \; [a = b]P \; \mid \; D\langle\tilde{b}\rangle$$

A tilde denotes a tuple. When the tilde is empty, the surrounding brackets () and $\langle\rangle$ will be omitted. $\mathbf{0}$ is the inactive process. An input-prefixed process $a(\tilde{b}).\, P$ waits for a tuple of names $\tilde{c}$ to be sent along a and then behaves like $P\{\tilde{c}/\tilde{b}\}$, where $\{\tilde{c}/\tilde{b}\}$ is the componentwise and parallel substitution of names $\tilde{b}$ with names $\tilde{c}$. An output-prefixed process $\overline{a}\langle\tilde{b}\rangle.\, P$ sends $\tilde{b}$ along a and then continues like P. Sum and parallel composition are used, as in CCS, to express a choice and to run two processes in parallel. The restriction $\boldsymbol{\nu}\, a\, P$ makes name a local, or private, to P. The *matching* construct $[a = b]P$ is used to test for syntactic equality of names; $[a = b]P$ behaves like P if a and b are the same name, as $\mathbf{0}$ otherwise. Constants are used to write recursive definitions of processes. A constant

D has a defining equation of the form $D \stackrel{def}{=} (\widetilde{c})P$, which can be thought as a procedure with formal parameters $\tilde{c}$; then $D\langle\widetilde{b}\rangle$ is like a procedure call with actual parameters $\widetilde{b}$.

In prefixes $a(\widetilde{b})$ and $\overline{a}\langle\widetilde{b}\rangle$, we call a the *subject* and $\widetilde{b}$ the *object*. We use α to range over prefixes and often abbreviate $\alpha.\mathbf{0}$ as α. We assign parallel composition and sum the lowest precedence among the operators. The operators $a(\widetilde{b}).P$, $(\widetilde{b})P$ and $\boldsymbol{\nu}\, b\, P$ bind all free occurrences of names $\widetilde{b}$ and b in P. These binders give rise in the expected way to the definition of *free names* of a term. The definitions of substitution and alpha conversion are standard too, with renaming possibly involved to avoid capture of free names. *We identify processes or actions which only differ on the choice of the bound names*; this will avoid some tedious side conditions, especially in the definitions of bisimulations. The symbol $=$ will mean "syntactic identity modulo alpha conversion". In a constant definition $D \stackrel{def}{=} (\widetilde{c})P$, it is assumed that $\widetilde{c}$ contains all names free in P. Sometimes, we use $\stackrel{def}{=}$ simply as abbreviation mechanism, to assign a name to an expression to which we want to refer later.

There are two derived operators that we sometimes use. One is the silent prefix $\tau.\, P$, representing a process which can evolve to P without requiring communications with the environment; it can be defined as $\boldsymbol{\nu}\, a\, (\overline{a} \mid a.P)$, if a is not free in P. The other is the replication $!\, P$, which in some presentations of π-calculus is chosen in place of recursive definitions, to express processes with infinite behaviour. Intuitively, $!\, P$ stands for a countable infinite number of copies of P in parallel. In the interleaving setting, recursion and replication are interdefinable, up to some weak equivalence [Mil91, section 3]. In a non-interleaving setting, it is still straightforward to code replication up using recursion; but the other way round may fail. In particular, it fails in our spatiality-based framework, as pointed out in Remark 4.7; this is the reason why we adopted recursion in the basic syntax.

2.2 Sorting

Virtually all π-calculus processes described in the literature obey some discipline in the use of names. The introduction of sorts and sortings into the π-calculus [Mil91] intends to make this name discipline explicit. In the polyadic π-calculus, sorts are also essential to avoid disagreement in the arities of tuples carried by a given name, or applied to a given constant. Below, we review the definition of sorting and of well-sorted process.

Names are partitioned into a collection of *subject sorts*, ranged over by s, each of which contains an infinite number of names. We write $a : s$ to mean that name a belongs to the subject sort s; this notation is extended to tuples componentwise. Then *object sorts*, ranged over by S, are just sequences of subject sorts, such as $(s_1, \ldots, s_n)$ or (s). Finally, a *sorting* is a function Ob mapping each subject sort onto an object sort. We write $s \mapsto (\tilde{s}) \in Ob$, if Ob assigns the object sort $(\tilde{s})$ to s. By assigning the object sort (s_1, s_2) to the subject sort s, one forces the tuples carried by any name in s to be a pair whose first component is a name of s_1 and whose second component is a name of s_2. Thus, if $a : s \mapsto (s_1, s_2)$, then for $a(\tilde{x}).P$ and $\overline{a}\langle\tilde{x}\rangle.P$ to respect Ob, it must be that $\tilde{x} = x_1, x_2$, for some $x_1 : s_1$ and $x_2 : s_2$. Moreover, a process respecting Ob must contain only matching $[a = b]$ between names a and b of the same sort, and it must have correct applications. To say what the latter means, we first assign an object sort to agents: Processes take sort $()$, whereas if $D \stackrel{def}{=} (\tilde{x})P$ with $\tilde{x} : \tilde{s}$, then D, and $(\tilde{x})P$ take sort $(\tilde{s})$. Now, the requirement on an application $D\langle\tilde{y}\rangle$ is that $\tilde{s}$ exists s.t. $\tilde{y} : \tilde{s}$ and $D : (\tilde{s})$. To sum up, a term is *well-sorted* for Ob if all its prefixes, matching and applications obey the discipline given by Ob, as described above. In the remainder of the paper, all processes are supposed to be well-sorted for some sorting Ob.

2.3 The standard labelled transition system and observation equivalence

There are three possible forms of action. A *silent action* $P \stackrel{\tau}{\longrightarrow} P'$ represents interaction, i.e. an internal activity in P (τ is a special symbol which does not appear in the set of all names). *Input* and *output actions* are, respectively, of the form

$$P \stackrel{a\langle\widetilde{b}\rangle}{\longrightarrow} P' \qquad \text{and} \qquad P \stackrel{(\boldsymbol{\nu}\,\widetilde{b'})\overline{a}\langle\widetilde{b}\rangle}{\longrightarrow} P'\,.$$

In both cases, the action occurs at a. In the input action, $\widetilde{b}$ is the tuple of names which are received. In the output action, $\widetilde{b}$ is the tuple of names which are emitted, and $\tilde{b'} \subseteq \widetilde{b}$ are private names which are carried out from their current scope (*scope extrusion*). Note the different brackets in input prefix and input action (round in the former, angled in the latter). This is to recall that in an input prefix $a(\widetilde{b})$, the names $\widetilde{b}$ are *binders* (waiting to be instantiated) whereas in an input action $a\langle\widetilde{b}\rangle$ they represent *values* (with which the binders have been instantiated) — in the same way as names $\widetilde{b}$ are values in an output $\overline{a}\langle\widetilde{b}\rangle$.

We use μ to represent the label of a generic action (not to be confused with α, which ranges over prefixes). All names in an input action are *free*. In an output action $(\boldsymbol{\nu}\,\widetilde{b'})\overline{a}\langle\widetilde{b}\rangle$, names $\widetilde{b'}$ are *bound*, the remaining ones free. Bound and free names of an action μ, respectively written $bn(\mu)$ and $fn(\mu)$, are defined accordingly. The *names* of μ, briefly $n(\mu)$, are $bn(\mu) \cup fn(\mu)$. In an output action $(\boldsymbol{\nu}\,\widetilde{b'})\overline{a}\langle\widetilde{b}\rangle$, the order of the names in $\widetilde{b'}$ is irrelevant, hence we identify two output actions $(\boldsymbol{\nu}\,\widetilde{b'})\overline{a}\langle\widetilde{b}\rangle$ and $(\boldsymbol{\nu}\,\widetilde{b''})\overline{a}\langle\widetilde{b}\rangle$ if $\widetilde{b'}$ and $\widetilde{b''}$ only differ on the order of their names.

The *standard transition system* of π-calculus is reported in Table 1 (rule S-LOC is not needed for standard processes, since they do not contain location prefixes). We have omitted the symmetric versions of rules S-SUM, S-PAR and S-COM. We work up to alpha conversion on processes also in transition systems, for which *alpha convertible agents are deemed to have the same transitions.* The rules for the derived operators of silent prefix and replication are:

$$\tau.\,P \xrightarrow{\tau} P \qquad\qquad \frac{P \mid !\,P \xrightarrow{\mu} P'}{!\,P \xrightarrow{\mu} P'}\,.$$

If a process P cannot immediately perform silent transitions, that is, there is no P' s.t. $P \xrightarrow{\tau} P'$, we say that P is *stable*; similarly, a sum $P + Q$ is stable if both summands P and Q are stable. We often abbreviate $P \xrightarrow{\tau} Q$ with $P \longrightarrow Q$. We write $P \xrightarrow{\widehat{\mu}} Q$ to mean:

- $P \xrightarrow{\mu} Q$, if $\mu \neq \tau$,
- $P = Q$ or $P \xrightarrow{\tau} Q$, if $\mu = \tau$.

As usual, the "weak" arrow $\Longrightarrow$ is the reflexive and transitive closure of $\longrightarrow$, that is $\Longrightarrow = \bigcup_{n\geq 0} \longrightarrow^n$; then $\stackrel{\mu}{\Longrightarrow}$ stands for $\Longrightarrow\xrightarrow{\mu}\Longrightarrow$. Finally, similarly to $P \xrightarrow{\widehat{\mu}} Q$, we use $P \stackrel{\widehat{\mu}}{\Longrightarrow} Q$ to mean:

- $P \stackrel{\mu}{\Longrightarrow} Q$, if $\mu \neq \tau$,
- $P \Longrightarrow Q$, if $\mu = \tau$.

Remark 2.1 [This is a technical remark on early transition systems and bisimulations which the non-interested reader may safely skip]

In the transition system of Table 1 the bound names of an input are instantiated as soon as possible, in the input rule; therefore it is an *early transition system* [San92], as opposed to a *late transition system* [MPW92, Mil91] in which the instantiation is done later, in the communication rule. The adoption of an early transition system naturally leads to the adoption of an *early bisimulation*, so christened in the literature to distinguish it from other formulations, like the *late* and the *open* [PS93, San93]. Our "early" choice is not critical for the results we shall present, which — we believe — can be adapted to late and open bisimulations. $\square$

$$\text{S-INP: } a(\widetilde{c}).\,X \xrightarrow{a\langle\widetilde{b}\rangle} X\{\widetilde{b}/\widetilde{c}\} \qquad \text{S-OUT: } \overline{a}\langle\widetilde{b}\rangle.\,X \xrightarrow{\overline{a}\langle\widetilde{b}\rangle} X$$

$$\text{S-SUM: } \frac{X \xrightarrow{\mu} X'}{X + Y \xrightarrow{\mu} X'} \qquad \text{S-PAR: } \frac{X \xrightarrow{\mu} X'}{X \mid Y \xrightarrow{\mu} X' \mid Y} \; bn(\mu) \cap fn(Y) = \emptyset$$

$$\text{S-COM: } \frac{X \xrightarrow{(\boldsymbol{\nu}\,\widetilde{b'})\overline{a}\langle\widetilde{b}\rangle} X' \qquad Y \xrightarrow{a\langle\widetilde{b}\rangle} Y'}{X \mid Y \xrightarrow{\tau} \boldsymbol{\nu}\,\widetilde{b'}\,(X' \mid Y')} \; \widetilde{b'} \cap fn(Y) = \emptyset$$

$$\text{S-RES: } \frac{X \xrightarrow{\mu} X'}{\boldsymbol{\nu}\,c\,X \xrightarrow{\mu} \boldsymbol{\nu}\,c\,X'} \; c \notin n(\mu) \qquad \text{S-OPEN: } \frac{X \xrightarrow{(\boldsymbol{\nu}\,\widetilde{b'})\overline{a}\langle\widetilde{b}\rangle} X'}{\boldsymbol{\nu}\,c\,X \xrightarrow{(\boldsymbol{\nu}\,\widetilde{b'}c)\overline{a}\langle\widetilde{b}\rangle} X'} \; c \neq a, c \in \widetilde{b} - \widetilde{b'}$$

$$\text{S-MATCH: } \frac{X \xrightarrow{\mu} X'}{[a = a]X \xrightarrow{\mu} X'} \qquad \text{S-CONS: } \frac{X\{\widetilde{b}/\widetilde{c}\} \xrightarrow{\mu} X'}{D\langle\widetilde{b}\rangle \xrightarrow{\mu} X'} \; D \stackrel{def}{=} (\widetilde{c})X$$

For located processes only: $\text{S-LOC}: \dfrac{X \xrightarrow{\mu} X'}{u :: X \xrightarrow{\mu} u :: X'}$

Table 1: The standard transitions

Observation equivalence. On the weak arrows we define observation equivalence, sometimes called weak bisimulation.[1]

Definition 2.2 (Observation equivalence) *A symmetric relation $\mathcal{R} \subseteq \mathcal{P} \times \mathcal{P}$ is a* weak bisimulation *(or a $\approx$-bisimulation) if $P \;\mathcal{R}\; Q$ and $P \stackrel{\mu}{\Longrightarrow} P'$ implies that there exists Q' s.t. $Q \stackrel{\widehat{\mu}}{\Longrightarrow} Q'$ and $P' \;\mathcal{R}\; Q'$. Two processes P and Q are* observation equivalent*, (or* weak bisimilar*), written $P \approx Q$, if $P \;\mathcal{R}\; Q$, for some weak bisimulation $\mathcal{R}$.* □

Proposition 2.3 *The relation $\approx$ is preserved by the operators of parallel composition, output prefix, stable sum, matching and restriction.* □

Observation equivalence is not preserved by non-stable sum and input prefix. As counterexample for the former, take processes $\tau.\,b.\,\mathbf{0}$ and $b.\,\mathbf{0}$ (the first is non-stable); their bisimilarity breaks down in the context $[\cdot] + a.\,\mathbf{0}$. The failure on input prefix arises

[1] Our definition of observation equivalence is formally the same as for CCS. This is possible because we work up to alpha conversion and because we have "free inputs" $P \xrightarrow{a\langle\widetilde{b}\rangle} Q$ in the transition system. The latter are the distinguishing feature of an early transition system and are not present in the grammar of π-calculus. Without free inputs, input actions would require a specific clause in the definition of bisimulation; see for instance [PS93].

because $\approx$ is not preserved by name instantiation. For instance, $[a = b]a.\,\mathbf{0} \approx \mathbf{0}$, but $([a = b]a.\,\mathbf{0})\{a/b\} \not\approx \mathbf{0}\{a/b\}$, since $([a = b]a.\,\mathbf{0})\{a/b\} = [a = a]a.\,\mathbf{0}$ is not a deadlocked process. In consequence, we also have $c(a).\,[a = b]a.\,\mathbf{0} \not\approx c(a).\,\mathbf{0}$.

We therefore consider also the *congruence* induced by $\approx$, called *observation congruence* [Mil89, MPW92].

Definition 2.4 (observation congruence) *Processes P and Q are* observation congruent*, written $P \approx^{c} Q$, if for all substitutions σ,*

1. *whenever $P\sigma \overset{\mu}{\Longrightarrow} P'$, there exists Q' s.t. $Q\sigma \overset{\mu}{\Longrightarrow} Q'$ and $P' \approx Q'$;*

2. *the converse, i.e.*
   whenever $Q\sigma \overset{\mu}{\Longrightarrow} Q'$, there exists P' s.t. $P\sigma \overset{\mu}{\Longrightarrow} P'$ and $P' \approx Q'$. □

The difference w.r.t. the definition of $\approx$ is the use of substitution and of the arrow $\overset{\mu}{\Longrightarrow}$ in place of $\overset{\widehat{\mu}}{\Longrightarrow}$.

Structural congruence. Roughly, two processes are structurally congruent if they may only differ syntactically with each other on the way in which their respective subcomponents are assembled. Recent work by Engelfriet [Eng93] has evidenced the link between structural congruence and a multiset semantics of π-calculus in which the meaning of a process is taken to be the multiset of its concurrent subprocesses. Structural congruence [Mil91] is written $\equiv$ and is defined as the smallest congruence over processes which satisfies the rules in Table 2. These are self-evident rules which must be valid in any reasonable behavioural equivalence. In this paper, $\equiv$ is useful to express the strength of certain operational correspondence results for our encodings, in cases in which the syntactic identity relation could not be used. The lemma below shows that structural congruence commutes with the transition relation.

Lemma 2.5 *If P and Q are standard processes with $P \equiv Q$, then $P \xrightarrow{\mu} P'$ implies $Q \xrightarrow{\mu} \equiv P'$.* □

Some laws for observation equivalence and congruence. Lemma 2.6 contains some laws for $\approx$ and $\approx^{c}$ which we shall need later; (1-3) are laws for the restriction operator; (4-6) are laws for parallel composition, obtained from the classical interleaving *expansion law* [Mil89, PS93] plus a few other simple laws; finally, (7-9) are distributivity laws for a restricted replication (for the proof of (9), see [Mil91] or [San92, Theorem 4.3.3]; the latter is proved in the Higher-Order π-calculus, of which the π-calculus is a special case). All these laws are valid for $\approx^{c}$, hence can be applied inside any context. Exceptions are laws (5) and (6) which, however, can be applied if the context contains no summation.

Abelian monoid laws for $+$:	$X_1 \mid X_2$	$\equiv$	$X_2 + X_1$
	$X_1 + (X_2 + X_3)$	$\equiv$	$(X_1 + X_2) + X_3$
	$X_1 + \mathbf{0}$	$\equiv$	X_1
Abelian monoid laws for $\mid$:	$X_1 \mid X_2$	$\equiv$	$X_2 \mid X_1$
	$X_1 \mid (X_2 \mid X_3)$	$\equiv$	$(X_1 \mid X_2) \mid X_3$
	$X \mid \mathbf{0}$	$\equiv$	X
Restriction:	$\boldsymbol{\nu}\, a\, \mathbf{0}$	$\equiv$	$\mathbf{0}$
	$\boldsymbol{\nu}\, a\, \boldsymbol{\nu}\, b\, X$	$\equiv$	$\boldsymbol{\nu}\, b\, \boldsymbol{\nu}\, a\, X$
if $a \notin fn(Y)$, then	$(\boldsymbol{\nu}\, a\, X) \mid Y$	$\equiv$	$\boldsymbol{\nu}\, a\, (X \mid Y)$
Matching:	$[a = a]X$	$\equiv$	X
Constant: if $D \stackrel{def}{=} (\tilde{c})P$ then	$D\langle \tilde{b} \rangle$	$\equiv$	$P\{\tilde{b}/\tilde{c}\}$
For located processes only:			
Location prefix:	$u :: (\boldsymbol{\nu}\, a\, X)$	$\equiv$	$\boldsymbol{\nu}\, a\, (u :: X)$
	$u :: (X_1 \mid X_2)$	$\equiv$	$(u :: X_1) \mid (u :: X_2)$
	$u :: \mathbf{0}$	$\equiv$	$\mathbf{0}$

Table 2: The rules of structural congruence ($\equiv$)

Lemma 2.6

1. $\boldsymbol{\nu}\, a\,(P_1 \,+\, P_2) \approx^{\mathrm{c}} \boldsymbol{\nu}\, a\, P_1 \,+\, \boldsymbol{\nu}\, a\, P_2$;

2. $\boldsymbol{\nu}\, a\, \alpha.P \approx^{\mathrm{c}} \alpha.\boldsymbol{\nu}\, a\, P$, *if a does not appear in α;*

3. $\boldsymbol{\nu}\, a\, \alpha.P \approx^{\mathrm{c}} \mathbf{0}$, *if a is the subject of α;*

4. $a(\widetilde{b}).\, P \mid c(\widetilde{d}).\, Q \approx^{\mathrm{c}} a(\widetilde{b}).\,(P \mid c(\widetilde{d}).\, Q) \,+\, c(\widetilde{d}).\,(a(\widetilde{b}).\, P \mid Q)$,
   if $\widetilde{b} \cap fn(c(\widetilde{d}).\, Q) = \emptyset$ and $\widetilde{d} \cap fn(a(\widetilde{b}).\, P) = \emptyset$;

5. $\boldsymbol{\nu}\, a\, \big(a(\widetilde{c}).\, Q \mid \overline{a}\langle\widetilde{b}\rangle.\, P\big) \approx Q\{\widetilde{b}/\widetilde{c}\} \mid P$, *if $a \notin fn(Q\{\widetilde{b}/\widetilde{c}\}, P)$;*

6. $\boldsymbol{\nu}\, a\, \big(\,!\, a(\widetilde{c}).\, R \mid \overline{a}\langle\widetilde{b}\rangle.\, P\big) \approx \boldsymbol{\nu}\, a\, \big(\,!\, a(\widetilde{c}).\, R \mid R\{\widetilde{b}/\widetilde{c}\} \mid P\big)$;

7. $\boldsymbol{\nu}\, a\, \big(\,!\, a(\widetilde{c}).\, R \mid \alpha.\, P\big) \approx^{\mathrm{c}} \alpha.\, \boldsymbol{\nu}\, a\, (\,!\, a(\widetilde{c}).\, R \mid P)$, *if a does not appear in α;*

8. $\boldsymbol{\nu}\, a\, \big(\,!\, a(\widetilde{c}).\, R \mid P\big) \approx^{\mathrm{c}} P$, *if $a \notin fn(P)$;*

9. *If a occurs free in P, Q and R only as subject of output prefixes, then*

   (a) $\boldsymbol{\nu}\, a\, \big(\,!\, a(\widetilde{c}).\, R \mid P \mid Q\big) \approx^{\mathrm{c}} \boldsymbol{\nu}\, a\, \big(\,!\, a(\widetilde{c}).\, R \mid P\big) \mid \boldsymbol{\nu}\, a\, \big(\,!\, a(\widetilde{c}).\, R \mid Q\big)$,

   (b) $\boldsymbol{\nu}\, a\, \big(\,!\, a(\widetilde{c}).\, R \mid\, !\, P\big) \approx^{\mathrm{c}}\, !\, \boldsymbol{\nu}\, a\, (\,!\, a(\widetilde{c}).\, R \mid P)$. $\Box$

3 The expansion relation and the "up-to" techniques

For the proof of the results in this paper, we appeal to the expansion relation and to various "up-to" techniques, which we introduce in the following two subsections.

3.1 The expansion relation

The expansion relation $\lesssim$ is an asymmetric variant of $\approx$ which allows us to count the number of τ-actions performed by the processes. Thus, $P \lesssim Q$ holds if $P \approx Q$ but also Q has at least as many τ-moves as P. In other words, $P \lesssim Q$ means that P is at least as fast as Q or, more generally, Q uses at least as many resources as P.

Definition 3.1 (expansion) *A relation $\mathcal{R} \subseteq \mathcal{P}\times\mathcal{P}$ is an* expansion *if $P \,\mathcal{R}\, Q$ implies:*

1. *Whenever $P \xrightarrow{\mu} P'$, there exists Q' s.t. $Q \overset{\mu}{\Longrightarrow} Q'$ and $P' \,\mathcal{R}\, Q'$;*

2. *whenever $Q \xrightarrow{\mu} Q'$, there exists P' s.t. $P \xrightarrow{\widehat{\mu}} P'$ and $P' \,\mathcal{R}\, Q'$.*

We say that Q expands *P, written $P \lesssim Q$, if $P\ \mathcal{R}\ Q$, for some expansion $\mathcal{R}$.* $\square$

Note that, in contrast with the other relations considered in the paper, $\lesssim$ is not symmetric. For instance, $P \lesssim \tau.P$, but $\tau.P \not\lesssim P$ since $\tau.P$ has to perform more τ-actions than P in order to mimic its actions. In CCS, $\lesssim$ has been studied — using a different terminology — by Arun-Kumar and Hennessy [AKH92], who have showed that $\lesssim$ is a mathematically tractable preorder, and by Milner and Sangiorgi [SM92], who have investigated the usefulness of $\lesssim$ in defining "weak bisimulations up-to". The proof of Proposition 3.2 is similar to the proof of the analogous result for $\approx$.

Proposition 3.2 *In π-calculus, $\lesssim$ is a preorder and is preserved by the operators of parallel composition, output prefix, stable sums, matching and restriction.* $\square$

Proposition 3.3 *The relation $\equiv$ is strictly included in $\lesssim$, and $\lesssim$ is strictly included in $\approx$.* $\square$

3.2 The "up-to" techniques

The "up-to" techniques allow us to reduce the size of a relation $\mathcal{R}$ to exhibit for proving bisimilarities. We consider two basic up-to techniques. We called the first *up-to $\mathcal{T}$* technique [Mil89, SM92]: Here, $\mathcal{T}$ is some well-known behavioural equivalence which is utilised to achieve the closure of $\mathcal{R}$. The other, called *up-to context* technique and introduced very recently in [San94], admits the use of contexts to achieve the closure of $\mathcal{R}$. At the end of the section we shall see that — at some extent — these two techniques can also be merged together, yielding a fairly powerful construction.

We mainly use the "up-to" techniques for proving $\approx$-bisimilarities. However, the "up-to $\equiv$" technique will also be applied to the expansion relation and to location bisimulation. The definitions of expansion up-to $\equiv$ and of location bisimulation up-to $\equiv$, and their soundness proofs are similar to the corresponding definition and proof of $\approx$-bisimulation up-to $\equiv$ (Theorem 3.5(2)). Given a relation $\mathcal{R}$ on processes, we denote by $\mathcal{R}^{-1}$ its inverse, i.e., $(P, Q) \in \mathcal{R}^{-1}$ iff $(Q, P) \in \mathcal{R}$.

$\approx$-bisimulation up-to $\mathcal{T}$. There are a few variants of $\approx$-bisimulation up-to $\mathcal{T}$ [SM92]. Here we only consider those which are needed in the paper.

Definition 3.4 ($\approx$-bisimulation up-to $\mathcal{T}$) *Let $\mathcal{T}$ and $\mathcal{R}$ be relation on $\mathcal{P} \times \mathcal{P}$, and $\mathcal{R}$ be symmetric. We say that $\mathcal{R}$ is a* $\approx$-bisimulation up-to $\mathcal{T}$ *if $P\ \mathcal{R}\ Q$ and $P \stackrel{\mu}{\Longrightarrow} P'$ imply that there exists Q' s.t. $Q \stackrel{\widehat{\mu}}{\Longrightarrow} Q'$ and $P' \mathcal{T}\, \mathcal{R}\, \mathcal{T}^{-1} Q'$.* $\square$

Theorem 3.5 *Suppose* $\mathcal{R}$ *is*

1. *a* $\approx$*-bisimulation up-to* $\equiv$ *or*

2. *a* $\approx$*-bisimulation up-to* $\approx$*;*

then $\mathcal{R} \subseteq \approx$.

PROOF: For (2) take

$$\mathcal{R}^{\approx} = \{(P, Q) \text{ s.t. } P \approx \mathcal{R} \approx Q\}.$$

The relation $\mathcal{R}^{\approx}$ is a weak bisimulation, as shown by the following diagram:

$$\begin{array}{ccccccccc}
P & \approx & P_1 & & \mathcal{R} & & Q_1 & \approx & Q \\
{\scriptstyle \mu}\Downarrow & & {\scriptstyle \widehat{\mu}}\Downarrow & & & & {\scriptstyle \widehat{\mu}}\Downarrow & & {\scriptstyle \widehat{\mu}}\Downarrow \\
P' & \approx & P'_1 & \approx & \mathcal{R} & \approx & Q'_1 & \approx & Q'
\end{array}$$

The proof of (1) is similar (to fill in the diagram one needs Lemma 2.5). $\Box$

$\approx$*-bisimulation up-to contexts.* For our purposes, a restricted form of $\approx$-bisimulation up-to context will be enough, in which only contexts of the form $\boldsymbol{\nu}\, \widetilde{c}\,(R \mid [\cdot])$ are employed. We call these *static contexts.*

Definition 3.6 ($\approx$-bisimulation up-to context) *A symmetric relation* $\mathcal{R}$ *is a* $\approx$-bisimulation up-to context *if* $P \mathrel{\mathcal{R}} Q$ *and* $P \stackrel{\mu}{\Longrightarrow} P''$ *imply that there are a static context* $C[\cdot]$ *and processes* P' *and* Q' *s.t.* $P'' = C[P']$, $Q \stackrel{\widehat{\mu}}{\Longrightarrow} C[Q']$ *and* $P' \mathrel{\mathcal{R}} Q'$. $\Box$

Therefore, the "up-to context" allows us to take advantage of a common context in the derivatives of P and Q, which can be canceled out. To show the soundness of the technique, we need a lemma.

Lemma 3.7 *Suppose* $\mathcal{R}$ *is a* $\approx$*-bisimulation up-to context,* $(P, Q) \in \mathcal{R}$ *holds and* $C[\cdot]$ *is a static context. If* $C[P] \xrightarrow{\mu_1} \cdots \xrightarrow{\mu_n} P_1$*,* $n \geq 0$*, then there are a static context* $C'[\cdot]$ *and processes* P' *and* Q' *s.t.* $P_1 \equiv C'[P']$*,* $C[Q] \stackrel{\widehat{\mu_1}}{\Longrightarrow} \cdots \stackrel{\widehat{\mu_n}}{\Longrightarrow} \equiv C'[Q']$ *and* $P' \mathrel{\mathcal{R}} Q'$*.*

PROOF: By induction on n. The basic case, when $n = 0$, is trivial. Suppose $n > 0$. If

$$C[P] \xrightarrow{\mu_1} \cdots \xrightarrow{\mu_{n-1}} P'_1,$$

then by induction there are a static context $C''[\cdot]$ and processes P'' and Q'' s.t.

$$P'_1 \equiv C''[P''] \ , \ C[Q] \stackrel{\widehat{\mu_1}}{\Longrightarrow} \cdots \stackrel{\widehat{\mu_{n-1}}}{\Longrightarrow} Q_1 \equiv C''[Q''] \text{ and } P'' \mathcal{R}\, Q''. \qquad (1)$$

Moreover, since $P'_1 \xrightarrow{\mu_n} P_1$, by Lemma 2.5, $C''[P''] \xrightarrow{\mu_n} P''_1 \equiv P_1$.

Let $C''[\cdot] \stackrel{def}{=} \boldsymbol{\nu}\,\widetilde{c}\,(R \mid [\cdot])$ and consider the action $\boldsymbol{\nu}\,\widetilde{c}\,(R \mid P'') \xrightarrow{\mu_n} P''_1$: There are three possible subcases, according to whether the action comes from the subcomponent P'' alone, from R alone or from an interaction between R and P''. We only consider the first, since the others can be handled in a similar way. We also assume that none of the restricted names in $\widetilde{c}$ is extruded; the case in which some of them is, can be easily accommodated. Suppose then $P'' \xrightarrow{\mu_n} P'''$. Since $P'' \mathcal{R}\, Q''$, for some static context $C'''[\cdot] \stackrel{def}{=} \boldsymbol{\nu}\,\widetilde{c'}\,(R' \mid [\cdot])$, and processes P' and Q', we have

$$P''' = C'''[P'] \qquad \text{and} \qquad Q'' \stackrel{\widehat{\mu_n}}{\Longrightarrow} Q''' = C'''[Q'] \text{ with } P' \mathcal{R}\, Q'.$$

Therefore we have

$$\begin{array}{lllll} C''[P''] & \xrightarrow{\mu_n} & P''_1 \ = & C''[P'''] & = \boldsymbol{\nu}\,\widetilde{c}\,(R \mid \boldsymbol{\nu}\,\widetilde{c'}\,(R' \mid P')) \ \equiv \boldsymbol{\nu}\,\widetilde{c}\widetilde{c'}\,\big((R \mid R') \mid P'\big) = C'[P'] \\ C''[Q''] & \stackrel{\widehat{\mu_n}}{\Longrightarrow} & & C''[Q'''] & = \boldsymbol{\nu}\,\widetilde{c}\,(R \mid \boldsymbol{\nu}\,\widetilde{c'}\,(R' \mid Q')) \ \equiv \boldsymbol{\nu}\,\widetilde{c}\widetilde{c'}\,\big((R \mid R') \mid Q'\big) = C'[Q'] \end{array}$$

where $C'[\cdot] \stackrel{def}{=} \boldsymbol{\nu}\,\widetilde{c}\widetilde{c'}\,\big((R|R')|[\cdot]\big)$. Further, from (1) and Lemma 2.5, also $Q_1 \stackrel{\widehat{\mu_n}}{\Longrightarrow} \equiv C'[Q']$. Summarising, we have obtained that

$$C[P] \xrightarrow{\mu_1} \cdots \xrightarrow{\mu_n} \equiv C'[P'], \qquad C[Q] \stackrel{\widehat{\mu_1}}{\Longrightarrow} \cdots \stackrel{\widehat{\mu_n}}{\Longrightarrow} \equiv C'[Q'] \qquad \text{and} \qquad P' \mathcal{R}\, Q'.$$

This proves the lemma. □

Theorem 3.8 *If $\mathcal{R}$ is a $\approx$-bisimulation up-to context, then $\mathcal{R} \subseteq \approx$.*

PROOF: Take

$$\mathcal{R}^{\mathrm{c}} = \{(C[P], C[Q]) \text{ s.t. } C[\cdot] \text{ is a static context and } P \mathcal{R}\, Q\}$$

We show that $\mathcal{R}^{\mathrm{c}}$ is a $\approx$-bisimulation up-to $\equiv$. This would prove the lemma, because it holds that $\mathcal{R} \subseteq \equiv \mathcal{R}^{\mathrm{c}} \equiv$. Suppose $C[P] \stackrel{\mu}{\Longrightarrow} P_1$. By Lemma 3.7 there are a static context $C'[\cdot]$ and processes P' and Q' s.t. $P_1 \equiv C'[P']$, $C[Q] \stackrel{\widehat{\mu}}{\Longrightarrow} \equiv C'[Q']$ and $P' \mathcal{R}\, Q'$. This closes up the bisimulation $\mathcal{R}^{\mathrm{c}}$. □

Bisimulation up-to context and up-to $\gtrsim$. In some cases, the up-to $\mathcal{T}$ and up-to context techniques can be combined together.

Definition 3.9 ($\approx$-bisimulation up-to context and up-to $\gtrsim$) *A symmetric relation $\mathcal{R} \subseteq \mathcal{P} \times \mathcal{P}$ is a* $\approx$-bisimulation up-to context and up-to $\gtrsim$ *if $P \mathcal{R} Q$ and $P \stackrel{\mu}{\Longrightarrow} P''$ imply that there are a static context $C[\cdot]$ and processes P' and Q' s.t. $P'' \gtrsim C[P']$, $Q \stackrel{\widehat{\mu}}{\Longrightarrow} \gtrsim C[Q']$ and $P' \mathcal{R} Q'$.*

In the above definition, the occurrence of $\gtrsim$ in $Q \stackrel{\widehat{\mu}}{\Longrightarrow} \gtrsim C[Q']$ could be weakened to $\approx$, and the soundness of the technique, namely Theorem 3.11, would still hold. However, we do not know whether the same weakening can be made on the requirement $P'' \gtrsim C[P']$.

Lemma 3.10 *Suppose $\mathcal{R}$ is a $\approx$-bisimulation up-to context and up-to $\gtrsim$, $(P, Q) \in \mathcal{R}$ holds and $C[\cdot]$ is a static context. If $C[P] \xrightarrow{\mu_1} \cdots \xrightarrow{\mu_n} P_1$, $n \geq 0$, then there are a static context $C'[\cdot]$ and processes P' and Q' s.t. $P_1 \gtrsim C'[P']$, $C[Q] \stackrel{\widehat{\mu_1}}{\Longrightarrow} \cdots \stackrel{\widehat{\mu_n}}{\Longrightarrow} \gtrsim C'[Q']$ and $P' \mathcal{R} Q'$.*

Proof: Similar to the proof of Lemma 3.7. $\square$

Theorem 3.11 *If $\mathcal{R}$ is a bisimulation up-to context and up-to $\gtrsim$, then $\mathcal{R} \subseteq \approx$.*

Proof: We show that

$$\mathcal{R}^{c,\approx} = \{(P,Q) \text{ s.t. } \begin{array}{l} \text{for some static context } C[\cdot] \text{ and proceses } P'', Q'' \\ \text{it holds that } P \gtrsim C[P'], Q \gtrsim C[Q'] \text{ and } P' \mathcal{R} Q' \quad \} \end{array}$$

is a $\approx$-bisimulation. Suppose $P \stackrel{\mu}{\Longrightarrow} P_1$. Since $C[P'] \lesssim P$, it holds that $C[P'] \stackrel{\widehat{\mu}}{\Longrightarrow} P_1' \lesssim P_1$. By Lemma 3.10 there are $C'[\cdot]$, P'' and Q'' s.t.

$$P_1' \gtrsim C'[P''], \quad C[Q'] \stackrel{\widehat{\mu}}{\Longrightarrow} \gtrsim C'[Q''] \quad \text{and} \quad P'' \mathcal{R} Q''.$$

Moreover, since $Q \gtrsim C[Q']$, for some Q_1 also $Q \stackrel{\widehat{\mu}}{\Longrightarrow} Q_1 \gtrsim C'[Q'']$ holds. Therefore, we have obtained that $P \stackrel{\mu}{\Longrightarrow} P_1 \gtrsim C'[P'']$, $Q \stackrel{\widehat{\mu}}{\Longrightarrow} Q_1 \gtrsim C'[Q'']$ and $P'' \mathcal{R} Q''$. This gives $P_1 \mathcal{R} Q_1$. $\square$

4 Location bisimulation

We now present location bisimulation for π-calculus. We shall follow what Boudol *et al.* have done for CCS in [BCHK91b]; their formalism adapts smoothly to the π-calculus, where name-passing does not introduce additional complications. The purpose of location bisimulation is to capture the spatial dependencies among processes. Different parallel components of a system are thought as residing at different locations, which can be observed when a visible transition occurs. This is expressed through *location transitions* of the form $\xrightarrow[u]{\mu}$, where μ is a standard action and u the location at which μ occurs. Intuitively, u represents the *access path* to the component performing the action. To accompany the presence of locations, the syntax of standard processes is enriched with a *location prefixing* $u :: -$, meaning that u is the location for the process in $-$. Locations are created at the moment in which an action is performed. The prefix order on location names represents the notion of *sublocality.* For example, in the process $a.(b \mid c)$, action a may occur at any location name, say l, and then b and c occur at sublocations of l, like lm and ln:

$$a.(b \mid c) \xrightarrow[l]{a} l :: (b \mid c) \xrightarrow[lm]{b} l :: (m :: \mathbf{0} \mid c) \xrightarrow[ln]{c} l :: (m :: \mathbf{0} \mid n :: \mathbf{0}) .$$

By contrast, silent transitions, since non observable, also have unobservable locations; consequently, they are defined using the standard transition system of the calculus. Locations, ranged over by $u, v, \ldots$, are words over the infinite alphabet $\mathcal{A}t$ of *atomic locations*, ranged over by l. We write uv for the concatenation of u and v and $\emptyset$ for the empty location. All notations and conventions introduced on standard processes — including the identification of alpha equivalent processes — extend to the enriched language of located processes below, and will not be repeated.

Definition 4.1 (Located processes) *The class $\mathcal{P}_{loc}$ of* located processes*, ranged over by $A, B ...$, is defined by the following grammar:*

$$A \;:=\; u :: A \;\mid\; A_1 \mid A_2 \;\mid\; \boldsymbol{\nu}\, a\, A \;\mid\; P \qquad \textit{where } P \textit{ is a standard process} \qquad \square$$

We do not allow the presence of locations underneath prefixes, matchings and sums or in constant definitions, because we are only interested in processes evolving from standard processes, for which these cases may never arise. This is clear from the rules for the *location transition system* in Table 3. The rules for the prefixes express the essence of the whole semantics: `L-INP` and `L-OUT`, for input and output prefixes, introduce locations into processes; `L-LOC`, for the location prefix $u :: A$, makes any action performed by A observed at a sublocation of u. The rules for the remaining operators are formally identical to the ordinary rules of Table 1. Note that if $A \xrightarrow[u]{\mu} A'$, then $\mu \neq \tau$. Indeed, silent actions do not exhibit locations, and are defined with the *standard transition system* of Table 1. Weak located transitions are defined in the usual manner, therefore

$$A \overset{\mu}{\underset{u}{\Longrightarrow}} A' \;\text{ is }\; A \Longrightarrow \xrightarrow[u]{\mu} \Longrightarrow A'.$$

$$\texttt{L-INP:}\quad a(\widetilde{c}).\,A \xrightarrow[l]{a\langle\widetilde{b}\rangle} l :: A\{\widetilde{b}/\widetilde{c}\},\quad l \in \mathcal{A}t \qquad \texttt{L-OUT:}\quad \overline{a}\langle\widetilde{b}\rangle.\,A \xrightarrow[l]{\overline{a}\langle\widetilde{b}\rangle} l :: A,\quad l \in \mathcal{A}t$$

$$\texttt{L-SUM:}\quad \frac{A \xrightarrow[u]{\mu} A'}{A + B \xrightarrow[u]{\mu} A'} \qquad \texttt{L-PAR:}\quad \frac{A \xrightarrow[u]{\mu} A'}{A \mid B \xrightarrow[u]{\mu} A' \mid B}\ bn(\mu) \cap fn(B) = \emptyset$$

$$\texttt{L-RES:}\quad \frac{A \xrightarrow[u]{\mu} A'}{\boldsymbol{\nu}\, c\, A \xrightarrow[u]{\mu} \boldsymbol{\nu}\, c\, A'}\ c \notin n(\mu) \qquad \texttt{L-OPEN:}\quad \frac{A \xrightarrow[u]{(\boldsymbol{\nu}\,\widetilde{b'})\overline{a}\langle\widetilde{b}\rangle} A'}{\boldsymbol{\nu}\, c\, A \xrightarrow[u]{(\boldsymbol{\nu}\,\widetilde{b'}c)\overline{a}\langle\widetilde{b}\rangle} A'}\ c \neq a, c \in \widetilde{b} - \widetilde{b'}$$

$$\texttt{L-MATCH:}\quad \frac{A \xrightarrow[u]{\mu} A'}{[a = a]A \xrightarrow[u]{\mu} A'} \qquad \texttt{L-CONS:}\quad \frac{A\{\widetilde{b}/\widetilde{c}\} \xrightarrow[u]{\mu} A'}{D\langle\widetilde{b}\rangle \xrightarrow[u]{\mu} A'}\ D \stackrel{def}{=} (\widetilde{c})A$$

$$\texttt{L-LOC:}\quad \frac{A \xrightarrow[v]{\mu} A'}{u :: A \xrightarrow[uv]{\mu} u :: A'}$$

Table 3: Location transitions

Using this location machinery, one can distinguish processes which have different degrees of parallelism, like $a \mid b$ and $a.\,b\, +\, b.\,a$: The actions of the former occur at independent locations, whereas those of the latter exhibit a nesting of locations (the two processes are observation congruent, by law (4) of Lemma 2.6). We can also distinguish processes with the same global, but not local, parallelism, like $a.\,\boldsymbol{\nu}\, c\,(b.\,c \mid \overline{c}.\,d)$ and $\boldsymbol{\nu}\, c\,(a.\,b.\,c \mid \overline{c}.\,d)$: In the former, the action at d is perceived at a sublocation of a's location, which is not the case in the latter. We write $\mathcal{A}t(A)$ for the set of atomic locations occurring in A.

Definition 4.2 (Location bisimulation) *A symmetric relation* $\mathcal{R} \subseteq \mathcal{P}_{loc} \times \mathcal{P}_{loc}$ *is a* location bisimulation *(or* $\approx_\ell$*-bisimulation) if* $A\,\mathcal{R}\,B$ *implies*

1. *whenever* $A \Longrightarrow A'$, *there exists* B' *s.t.* $B \Longrightarrow B'$ *and* $A'\,\mathcal{R}\,B'$;
2. *whenever* $A \overset{\mu}{\underset{ul}{\Longrightarrow}} A'$ *and* $l \notin \mathcal{A}t(A, B)$ *there exists* B' *s.t.* $B \overset{\mu}{\underset{ul}{\Longrightarrow}} B'$ *and* $A'\,\mathcal{R}\,B'$.

Two located processes A *and* B *are* location bisimilar, *written* $A \approx_\ell B$, *if* $A\,\mathcal{R}\,B$, *for some location bisimulation* $\mathcal{R}$ *.* □

In clause (2), we have followed Kiehn [Kie91] in requiring that l is new. This condition is not present in the original definition in [BCHK91b]. However the two formulations are equivalent, as a consequence of the lemma below.

Lemma 4.3 *Let A and B be located processes, and ρ a renaming of atomic locations. Then $A \approx_\ell B$ implies $A\rho \approx_\ell B\rho$.* $\square$

Location bisimulation is a (strict) refinment of observation equivalence.

Proposition 4.4 *If P and Q are standard processes, then $P \approx_\ell Q$ implies $P \approx Q$.* $\square$

We define the structural congruence relation for located processes as for standard processes, but with the addition of certain rules to push location prefixes inside parallel compositions and restrictions. See Table 2. Lemma 4.5 generalises Lemma 2.5 to located processes and shows commutativity of structural congruence with the standard and location transitions.

Lemma 4.5 *Let A and B be located processes with $A \equiv B$. We have:*

1. *If $A \xrightarrow{\mu} A'$, then $B \xrightarrow{\mu} \equiv A'$;*
2. *if $A \xrightarrow[u]{\mu} A'$, then $B \xrightarrow[u]{\mu} \equiv A'$.* $\square$

Lemma 4.6 *If P is a standard process and $P \xrightarrow[u]{\mu} A$, then u is an atomic location.* $\square$

Remark 4.7 It is easy to see why replication cannot code constants, i.e. recursive definitions of agents. Using the latter, we can define $D \stackrel{def}{=} (c)\overline{c}.\, D\langle c\rangle$. Process $D\langle a\rangle$ can perform actions at a with unbounded access-path length, like

$$D\langle a\rangle \xrightarrow[l_1]{\overline{a}} \xrightarrow[l_1 l_2]{\overline{a}} \cdots \xrightarrow[l_1\ldots l_n]{\overline{a}} \cdots.$$

This behaviour cannot be mimicked with replication in place of constants. $\square$

5 The Encoding of location bisimulation

In this section we present the encoding $\mathcal{S}$ which allows us to reconduct the *non-interleaving* location bisimulation to the ordinary *interleaving* observation equivalence. $\mathcal{S}$ is defined on *standard* processes. But to prove its correctness, we shall have to extend it to an encoding $\mathcal{L}$ on *located* processes.

First, we give some intuition on how $\mathcal{S}$ and $\mathcal{L}$ work. Suppose that $A \xrightarrow[ul]{a\langle\widetilde{b}\rangle} A'$. Then approximately, for any x we have

$$\mathcal{L}[\![A]\!] \xrightarrow{a\langle\widetilde{b}x\rangle} \equiv \boldsymbol{\nu}\, z\,(\mathrm{Loc}\langle xzy\rangle \mid \mathcal{L}[\![A']\!])\,. \tag{2}$$

Process $\mathrm{Loc}\langle xzy\rangle$ implements the location mechanism and can be explored by the external observer using name x. The locations in A are represented by special names in $\mathcal{L}[\![A]\!]$, called *locating names*. The correspondence between locations and locating names is created step-by-step as the computation proceeds. So in (2), y is the locating name which represented the location u of A in $\mathcal{L}[\![A]\!]$; and z is the locating name which will represent the "new" location ul of the derivative A' in $\mathcal{L}[\![A']\!]$. The constant Loc is so defined:

$$\mathrm{Loc} \stackrel{def}{=} (xzy)\big(\overline{x}\langle z\rangle \mid {!}\, z.\, \overline{y}\big)\,. \tag{3}$$

Thus, in (2) name z can be accessed using x; then, since an input at z triggers an output at the locating name y, we understand that y represents the location father of z. The presence of the component $\mathrm{Loc}\langle xyz\rangle$ in (2) does not complicate the verification of process bisimilarities: In the definition of location bisimulation, two actions

$$A \xrightarrow[u]{a\langle\widetilde{b}\rangle} A' \qquad\qquad \text{and} \qquad\qquad B \xrightarrow[v]{a\langle\widetilde{b}\rangle} B'$$

match with each other iff $u = v$ and A' is location bisimilar with B'. Correspondingly, using the encoding and observation equivalence, the two actions

$$\mathcal{L}[\![A]\!] \xrightarrow{a\langle\widetilde{bx}\rangle} \equiv \boldsymbol{\nu}\, z\,(\mathrm{Loc}\langle xzy\rangle \mid \mathcal{L}[\![A']\!]) \qquad \text{and} \qquad \mathcal{L}[\![B]\!] \xrightarrow{a\langle\widetilde{bx}\rangle} \equiv \boldsymbol{\nu}\, z\,(\mathrm{Loc}\langle xzy'\rangle \mid \mathcal{L}[\![B']\!])$$

match with each other iff $y = y'$and $\mathcal{L}[\![A']\!]$ is observation equivalent with $\mathcal{L}[\![B']\!]$. (This is a kind of cancellation property, which will be proved in Lemma 8.3).

Table 4 presents the encoding $\mathcal{S}$ for standard processes. It acts as a homomorphism everywhere but on prefixes. Since the arities of constants and of names are modified by $\mathcal{S}$ — an extra component has to be added — it has to be defined on sortings too. Names x, y and z, used to implement the location mechanism, are all different from each other and are drawn from two special new sorts s and t, to keep them separate from the names in the source processes. The names of the same sort as y and z are the locating names. Note that constant Loc has sort (s, t, t): Therefore, in the following, in all applications $\mathrm{Loc}\langle xyz\rangle$, name x is intended to belong to sort s, and y, z to sort t. The encoding also acts as a homomorphism on the derived operators of silent prefix and replication (for the former, up-to $\approx$):

$$\mathcal{S}[\![\tau.\, P]\!]\langle y\rangle \;\stackrel{def}{=}\; \tau.\, \mathcal{S}[\![P]\!]\langle y\rangle \qquad\qquad \mathcal{S}[\![\,!\, P]\!]\langle y\rangle \;\stackrel{def}{=}\; !\,\mathcal{S}[\![P]\!]\langle y\rangle\,.$$

It is straightforward to see that there is agreement between the definitions of $\mathcal{S}$ on processes and on sortings:

Proposition 5.1 *If $P :: Ob$, then $\mathcal{S}[\![P]\!]\langle y\rangle :: \mathcal{S}[\![Ob]\!]$, for any locating name y.* $\square$

Encoding of the sorting.

$$\begin{aligned}\mathcal{S}[\![Ob]\!] &\stackrel{def}{=} \{r \mapsto (\tilde{r}, s) \text{ s.t. } r \mapsto (\tilde{r}) \in Ob\} \;\bigcup \\ &\quad\;\{s \mapsto (t),\; t \mapsto ()\}\end{aligned}$$

where s and t are new sorts, i.e. sorts which do not appear in Ob.

Encoding of standard processes; we assume $x : s$ and $y, z : t$.

$$\begin{aligned}
\mathcal{S}[\![a(\tilde{b}).\,P]\!]\langle y\rangle &\stackrel{def}{=} a(\tilde{b}x).\,\boldsymbol{\nu}\,z\,\big(\mathrm{Loc}\langle xzy\rangle \mid \mathcal{S}[\![P]\!]\langle z\rangle\big)\\
\mathcal{S}[\![\overline{a}\langle\tilde{b}\rangle.\,P]\!]\langle y\rangle &\stackrel{def}{=} \boldsymbol{\nu}\,x\,\overline{a}\langle\tilde{b}x\rangle.\,\boldsymbol{\nu}\,z\,\big(\mathrm{Loc}\langle xzy\rangle \mid \mathcal{S}[\![P]\!]\langle z\rangle\big)
\end{aligned}$$

$$\begin{array}{rclrcl}
\mathcal{S}[\![P \mid Q]\!]\langle y\rangle &\stackrel{def}{=}& \mathcal{S}[\![P]\!]\langle y\rangle \mid \mathcal{S}[\![Q]\!]\langle y\rangle & \mathcal{S}[\![P + Q]\!]\langle y\rangle &\stackrel{def}{=}& \mathcal{S}[\![P]\!]\langle y\rangle + \mathcal{S}[\![Q]\!]\langle y\rangle\\
\mathcal{S}[\![\boldsymbol{\nu}\,a\,P]\!]\langle y\rangle &\stackrel{def}{=}& \boldsymbol{\nu}\,a\,\mathcal{S}[\![P]\!]\langle y\rangle & \mathcal{S}[\![\mathbf{0}]\!]\langle y\rangle &\stackrel{def}{=}& \mathbf{0}\\
\mathcal{S}[\![[a=b]P]\!]\langle y\rangle &\stackrel{def}{=}& [a=b]\mathcal{S}[\![P]\!]\langle y\rangle & \mathcal{S}[\![D\langle\tilde{b}\rangle]\!]\langle y\rangle &\stackrel{def}{=}& D\langle\tilde{b}y\rangle
\end{array}$$

The constant Loc is defined as $\mathrm{Loc} \stackrel{def}{=} (xzy)\big(\overline{x}\langle z\rangle \mid !\,z.\,\overline{y}\big)$.
Each equation $D \stackrel{def}{=} (\tilde{c})P$ is replaced by the equation $D \stackrel{def}{=} (\tilde{c}y)\mathcal{S}[\![P]\!]\langle y\rangle$.

Table 4: The encoding $\mathcal{S}$

More difficult is to show the semantic correctness of $\mathcal{S}$, i.e. $P \approx_\ell Q$ iff $\mathcal{S}[\![P]\!]\langle y\rangle \approx \mathcal{S}[\![Q]\!]\langle y\rangle$. The proof of this result will absorb a consistent part of the paper. The next example shows how the simulation of visible transitions of P is achieved by $\mathcal{S}[\![P]\!]$. Silent transitions will be examined after presenting the extension $\mathcal{L}$ of $\mathcal{S}$ to located processes. Then, using the examples given, we shall be able to supply further justifications for the peculiar structure of the component $\mathrm{Loc}\langle xzy\rangle$ employed in the encoding.

Example 5.2 *Let $P_1 \stackrel{def}{=} \overline{a}.P_2$ and $P_2 \stackrel{def}{=} \overline{b}.\mathbf{0}$. Then we have:*

$$P_1 \xrightarrow[l_1]{\overline{a}} l_1 :: P_2 \xrightarrow[l_1 l_2]{\overline{b}} l_1 l_2 :: \mathbf{0}$$

We understand that action $\overline{b}$ fired at a sublocation of a's location from the fact that l_1 is a prefix of $l_1 l_2$. Let us see how we can recover the same information from $\mathcal{S}$. We have

$$\begin{aligned}
\mathcal{S}[\![P_1]\!]\langle z\rangle &\equiv \boldsymbol{\nu}\,x_a\,\overline{a}\langle x_a\rangle.\,\boldsymbol{\nu}\,z_a\,\big(\overline{x_a}\langle z_a\rangle \mid !\,z_a.\,\overline{z} \mid \mathcal{S}[\![P_2]\!]\langle z_a\rangle\big)\\
\mathcal{S}[\![P_2]\!]\langle z_a\rangle &\equiv \boldsymbol{\nu}\,x_b\,\overline{b}\langle x_b\rangle.\,\boldsymbol{\nu}\,z_b\,(\overline{x_b}\langle z_b\rangle \mid !\,z_b.\,\overline{z_a} \mid \mathbf{0})
\end{aligned}$$

Process $\mathcal{S}[\![P_1]\!]\langle z\rangle$ may evolve as follows

$$\begin{array}{rcl} \mathcal{S}[\![P_1]\!]\langle z\rangle & \xrightarrow{(\boldsymbol{\nu}\, x_a)\overline{a}\langle x_a\rangle} \xrightarrow{(\boldsymbol{\nu}\, z_a)\overline{x_a}\langle z_a\rangle} \equiv & !\, z_a.\, \overline{z} \mid \mathcal{S}[\![P_2]\!]\langle z_a\rangle \\ & \xrightarrow{(\boldsymbol{\nu}\, x_b)\overline{b}\langle x_b\rangle} \xrightarrow{(\boldsymbol{\nu}\, z_b)\overline{x_b}\langle z_b\rangle} \equiv & !\, z_a.\, \overline{z} \mid !\, z_b.\, \overline{z_a} \mid \mathbf{0} \end{array}$$

Intuitively, z_a and z_b represent l_1 and $l_1 l_2$, respectively, and z represents the empty access path $\emptyset$. Interrogations at z_a are answered at z, revealing that the action at a occurred at the outermost level. By contrast, interrogations at z_b can be answered at z_a, revealing the sublocation relation between l_1 and $l_1 l_2$. Note that the simulation of each visible transition sets free a replicated process of the form $!\, y.\, \overline{y'}$. This is not a serious drawback since, as mentioned in the introduction of this section, such replicated processes can be canceled in the verification of process bisimilarities (see Lemma 8.2).

The encoding $\mathcal{S}$ for standard processes is extended to an encoding $\mathcal{L}$ for located processes in Table 5. Intuitively, there is a one-to-one correspondence between the locations in A and the locating names which are free in the process encoding A. We keep the record of this correspondence in a *locating function* φ, which is a finite injective function from locating names to locations; the adjective "finite" indicates that φ is only defined on a finite subset of locating names. Thus, if $\{y_1, \ldots, y_n\}$ is the set of names on which φ is defined, and $\varphi(y_i) = u_i$, $1 \leq i \leq n$, then we sometimes write $\varphi = \{(y_1, u_n), \ldots, (y_n, u_n)\}$. Being φ injective, we can also define its inverse φ^{-1}. If $\big(P, \varphi\big)$ encodes A, then φ must be defined on all locating names free in P. However φ is not unique, since we also want to allow φ to be defined on a superset of such names; for this reason we write $\big(P, \varphi\big) \in \mathcal{L}[\![A]\!]$, rather than $\big(P, \varphi\big) = \mathcal{L}[\![A]\!]$. We set:

$$\varphi_{-u} \stackrel{def}{=} \{(z, v) \text{ s.t. } (z, uv) \in \varphi\}\,.$$

That is, φ_{-u} is obtained from those elements of φ whose location component has u as prefix, by removing prefix u. Since φ is injective, φ_{-u} is injective too.

For each A, $\mathcal{L}[\![A]\!]$ is non-empty. Moreover, when encoding a set of located processes, it is possible to use the same locating function:

Lemma 5.3 *Let $A_1, \ldots, A_n$ be located processes, $n > 0$. Then there is a locating function φ and standard processes $P_1, \ldots, P_n$ s.t. for all i, $\big(P_i, \varphi\big) \in \mathcal{L}[\![A_i]\!]$.*

PROOF: Take all possible nesting of atomic locations as they appear in $A_1, \ldots, A_n$, and call $u_1, \ldots, u_m$ the locations so obtained. Take m different names $z_1, \ldots, z_m$ and define φ as $\{(z_j, u_j) \text{ s.t. } 1 \leq j \leq m\}$. Then use structural induction and the rules in Table 5 to find P_i s.t. $\big(P_i, \varphi\big) \in \mathcal{L}[\![A_i]\!]$, $1 \leq i \leq n$. $\square$

$$
\begin{array}{rcll}
\mathcal{L}[\![u :: A]\!] & \ni & \big(P, \varphi\big) & \text{if } \mathcal{L}[\![A]\!] \ni \big(P, \varphi_{-u}\big) \\
\mathcal{L}[\![A_1 \mid A_2]\!] & \ni & \big(P_1 \mid P_2, \varphi\big) & \text{if } \mathcal{L}[\![A_i]\!] \ni \big(P_i, \varphi\big), i = 1, 2 \\
\mathcal{L}[\![\boldsymbol{\nu}\, a\, A]\!] & \ni & \big(\boldsymbol{\nu}\, a\, P, \varphi\big) & \text{if } \mathcal{L}[\![A]\!] \ni \big(P, \varphi\big) \\
\mathcal{L}[\![P]\!] & \ni & \big(\mathcal{S}[\![P]\!]\langle y\rangle, \varphi\big) & \text{if } \varphi^{-1}(\emptyset\,) = y
\end{array}
$$

Table 5: The extension $\mathcal{L}$ of $\mathcal{S}$ to located processes

The example below shows how silent transitions of a located process are modelled with $\mathcal{L}$. In location bisimulation, a silent transition does not carry a location. In the encoding, we have to make sure that silent transitions do not disrupt the pointer machinery used to represent the location informations. This was a major obstacle to overcome in the definitions of $\mathcal{S}$ and $\mathcal{L}$.

Example 5.4 *Take* $A \stackrel{def}{=} u :: \overline{b} \mid v :: b.c$ *and consider the transitions*

$$
A \longrightarrow \equiv v :: c \xrightarrow[vl]{c} \equiv \mathbf{0} \tag{4}
$$

Action c is observed as occurring at the top level of the site denominated v. Let us see what happens with the encoding. If $(z,u),(y,v) \in \varphi$, then $\big(\mathcal{S}[\![\overline{b}]\!]\langle z\rangle \mid \mathcal{S}[\![b.c]\!]\langle y\rangle, \varphi\big) \in \mathcal{L}[\![A]\!]$ and we have:

$$
\begin{array}{rcl}
\mathcal{S}[\![\overline{b}]\!]\langle z\rangle \mid \mathcal{S}[\![b.c]\!]\langle y\rangle & = & \\
\big(\boldsymbol{\nu}\, x_b\, \overline{b}\langle x_b\rangle.\, \boldsymbol{\nu}\, z_b\, (\mathrm{Loc}\langle x_b z_b z\rangle \mid \mathbf{0})\big) \mid b(x'_b).\, \boldsymbol{\nu}\, z'_b \left(\mathrm{Loc}\langle x'_b z'_b y\rangle \mid \mathcal{S}[\![c.\mathbf{0}]\!]\langle z'_b\rangle\right) & \longrightarrow\equiv & \\
\boldsymbol{\nu}\, x_b \left(\boldsymbol{\nu}\, z_b\, \mathrm{Loc}\langle x_b z_b z\rangle \mid \boldsymbol{\nu}\, z'_b \left(\mathrm{Loc}\langle x_b z'_b y\rangle \mid \mathcal{S}[\![c.\mathbf{0}]\!]\langle z'_b\rangle\right)\right) & \stackrel{def}{=} & Q'
\end{array} \tag{5}
$$

which, expanding the definition of the constant Loc *gives*

$$
Q' \;\equiv\; \boldsymbol{\nu}\, x_b \left(\boldsymbol{\nu}\, z_b \left(\overline{x_b}\langle z_b\rangle \mid !\, z_b.\, \overline{z}\right) \mid \boldsymbol{\nu}\, z'_b \left(\overline{x_b}\langle z'_b\rangle \mid !\, z'_b.\, \overline{y} \mid \mathcal{S}[\![c.\mathbf{0}]\!]\langle z'_b\rangle\right)\right) \tag{6}
$$

The restrictions on x_b and z'_b cause deadlock in processes $\overline{x_b}\langle z_b\rangle \mid !\, z_b.\, \overline{z}$ and $\overline{x_b}\langle z'_b\rangle$; hence simple algebraic manipulations give

$$
\begin{array}{rl}
\approx & \boldsymbol{\nu}\, z'_b \left(!\, z'_b.\, \overline{y} \mid \mathcal{S}[\![c.\mathbf{0}]\!]\langle z'_b\rangle\right) \\
= & \boldsymbol{\nu}\, z'_b \left(!\, z'_b.\, \overline{y} \mid c(x).\, \boldsymbol{\nu}\, z_c \left(\overline{x}\langle z_c\rangle \mid !\, z_c.\, \overline{z'_b}\right)\right)
\end{array} \tag{7}
$$

Using the distributivity properties of restricted replications (laws (7-9) of Lemma 2.6), we get

$$\approx \quad c(x).\, \boldsymbol{\nu}\, z_c \left(\overline{x}\langle z_c\rangle \mid !\, z_c.\, \boldsymbol{\nu}\, z'_b\, (\, !\, z'_b.\, \overline{y} \mid \overline{z'_b})\right).$$

Finally, since $\boldsymbol{\nu}\, z'_b\, (\, !\, z'_b.\, \overline{y} | \overline{z'_b}) \approx \overline{y}$*, and this bisimilarity is preserved by contexts containing no summation (Lemma 2.6(6,8) and previous discussion), we get*

$$\approx \quad c(x).\, \boldsymbol{\nu}\, z_c \left(\overline{x}\langle z_c\rangle \mid !\, z_c.\, \overline{y}\right) = c(x).\, \boldsymbol{\nu}\, z_c \left(\mathrm{Loc}\langle x z_c y\rangle\right) = \mathcal{S}[\![c.\mathbf{0}]\!]\langle y\rangle\,. \tag{8}$$

Summarising, we have

$$\mathcal{S}[\![P]\!]\langle z\rangle \quad \longrightarrow \approx \quad \mathcal{S}[\![c.\mathbf{0}]\!]\langle y\rangle$$

and now

$$\mathcal{S}[\![c.\mathbf{0}]\!]\langle y\rangle = c(x).\, \boldsymbol{\nu}\, z_c \left(\overline{x}\langle z_c\rangle \mid !\, z_c.\, \overline{y}\right) \quad \xrightarrow{c\langle x\rangle} \; \xrightarrow{(\boldsymbol{\nu}\, z_c)\overline{x}\langle z_c\rangle} \equiv \quad !\, z_c.\, \overline{y}\,.$$

The action at c *is correctly perceived as occurring at the top level as it was in (4), since inputs at* z_c *are sent back at* y*, and* y *stands for location* v *(i.e., it was* $(y, v) \in \varphi$*).*

Let us derive a corollary of the transformations above, which we shall use later, in Example 10.1. Suppose we had had the restriction on b *at the outermost level, resulting in process*

$$\boldsymbol{\nu}\, b\, (\mathcal{S}[\![\overline{b}]\!]\langle z\rangle \mid \mathcal{S}[\![b.c]\!]\langle y\rangle)\,.$$

If in (5) we apply the rules for structural congruence to pull the restriction at x_b *at the outermost level, and then use law (5) of Lemma 2.6, we can derive the equality*

$$\boldsymbol{\nu}\, b\, (\mathcal{S}[\![\overline{b}]\!]\langle z\rangle \mid \mathcal{S}[\![b.c]\!]\langle y\rangle) \approx Q'.$$

This, together with $Q' \approx \mathcal{S}[\![c.\mathbf{0}]\!]\langle z\rangle$*, derived in (6-8) above, yields an equational proof of*

$$\boldsymbol{\nu}\, b\, (\mathcal{S}[\![\overline{b}]\!]\langle z\rangle \mid \mathcal{S}[\![b.c]\!]\langle y\rangle) \approx \mathcal{S}[\![c.\mathbf{0}]\!]\langle z\rangle\,. \tag{9}$$

We can now explain the presence of the replication and of the particle $\overline{x}\langle z\rangle$ in the component

$$\mathrm{Loc}\langle xzy\rangle = \overline{x}\langle z\rangle \mid !\, z.\, \overline{y} \tag{10}$$

used for the encoding of prefixes. Replication is needed to have the equation

$$\boldsymbol{\nu}\, z\, (\, !\, z.\, \overline{y} \mid \mathcal{S}[\![P]\!]\langle z\rangle) \approx \mathcal{S}[\![P]\!]\langle z\rangle\{y/z\} \tag{11}$$

of which transformation (7-8) of the previous example is a special case. (Actually, for our proofs it will be useful to be a little more precise in (11), and replace observation equivalence ($\approx$) with the expansion relation ($\gtrsim$) — see Proposition 6.2.)

On the other hand, the particle $\overline{x}\langle z\rangle$ of (10) is used to access z. One might wonder why the access to z must be obtained through x and not *directly*, i.e. adopting the following simplifications of the rules for prefixes:

$$\mathcal{S}[\![a(\widetilde{b}).\,P]\!]\langle y\rangle \stackrel{def}{=} a(\widetilde{b}z).\,\big(\,!\,z.\,\overline{y}\mid\mathcal{S}[\![P]\!]\langle z\rangle\big) \tag{12}$$

$$\mathcal{S}[\![\overline{a}\langle\widetilde{b}\rangle.\,P]\!]\langle y\rangle \stackrel{def}{=} \boldsymbol{\nu}\,z\,\overline{a}\langle\widetilde{b}z\rangle.\,\big(\,!\,z.\,\overline{y}\mid\mathcal{S}[\![P]\!]\langle z\rangle\big) \tag{13}$$

The problem with (12) and (13) is in the modeling of silent transitions. Rules (12) and (13) give rise to chains of pointers among locating names which are significantly different from those created by our original rules. Indeed, the adoption of (12) and (13) would lead us closer to *causal bisimulation* [DD89]. For instance, in Example 5.4, the reader might want to check that using rules (12) and (13) we would have

$$\mathcal{S}[\![\overline{b}]\!]\langle y\rangle\mid\mathcal{S}[\![b.c]\!]\langle z\rangle \stackrel{c\langle z_c\rangle}{\Longrightarrow} \approx \boldsymbol{\nu}\,z_b\,\big(\,!\,z_b.\,\overline{y}\mid\,!\,z_b.\,\overline{z}\mid\,!\,z_c.\,\overline{z_b}\big) \stackrel{def}{=} R$$

In R, each input at z_c can be answered back either by y or by z. This, intuitively, says that the causes of c are the union of the causes of the two b's whose interaction sets c free, which is precisely the philosophy of causal bisimulation.

6 Wire processes

Consider the process

$$W \stackrel{def}{=} !\,z(\widetilde{w}).\,\overline{y}\langle\widetilde{w}\rangle\,.$$

In an interleaving semantics, we might want to use W as a "wire" in $\boldsymbol{\nu}\,z\,(W\mid P)$ with endpoints z and y, so to transform every output by P at z into an output at y, and get the equation

$$\boldsymbol{\nu}\,z\,(W\mid P)\gtrsim P\{y/z\}\,.$$

To achieve this, we have to specify some constraints on the utilisation of z in P. Firstly, P should only perform outputs at z, and should not export it. Secondly, in the syntax of P we need to forbid certain uses of z within the dynamic operators of sum and prefixing. For instance, if $P \stackrel{def}{=} \overline{z}\langle w\rangle.\,a$, then $\boldsymbol{\nu}\,z\,(W\mid P)$ can perform actions $\overline{y}\langle w\rangle$ and a in either order, whereas $P\{y/z\}$ has to perform $\overline{y}\langle w\rangle$ before a. Similarly, if $P \stackrel{def}{=} \overline{z}\langle w\rangle + a$, then $\boldsymbol{\nu}\,z\,(W\mid P)\longrightarrow\boldsymbol{\nu}\,z\,(\overline{y}\langle w\rangle\mid W\mid\mathbf{0})\approx\overline{y}\langle v\rangle$, but $P\{y/z\}$ has no corresponding to such an evolution.

The solution we utilise is based on the following notion of "sender end-point". We say that a name z acts as a *sender end-point* in P if z occurs free in P only in

- subexpressions of the form $\overline{z}\langle\widetilde{w}\rangle.\,\mathbf{0}$ with $z \notin \{\widetilde{w}\}$, and which are not underneath a summation, or
- subexpressions of the form $\alpha.\overline{z}\langle\widetilde{w}\rangle.\,\mathbf{0}$ with $z \notin n(\alpha) \cup \{\widetilde{w}\}$.

For instance, z acts as a sender end-point in $(a.\,\overline{z}\langle w\rangle.\,\mathbf{0} + b.\mathbf{0}) \mid \overline{z}\langle w\rangle.\,\mathbf{0}$. We show that if z acts as a sender end-point in P, then W does behave as a wire in $\boldsymbol{\nu}\, z\,(W \mid P)$. To be able to use this result with our encoding $\mathcal{L}$, it is convenient to extend the grammar of our standard processes $\mathcal{P}$ with the replicator operator. We call $\mathcal{P}^*$ the resulting class of processes. We also add a rule for replication to those of structural congruence in Table 2:

$$!\,P \equiv P \mid !\,P\,.$$

Lemma 6.1 *Suppose that $P \in \mathcal{P}^*$ and that z acts as a sender end-point in P.*

1. *If $P\{y/z\} \xrightarrow{\mu} P''$ and $\mu \neq \tau$, then $P \xrightarrow{\mu'} P'$ with $\mu = \mu'\{y/z\}$ and $P'' = P'\{y/z\}$, for some P' and μ'.*
2. *If $P\{y/z\} \xrightarrow{\tau} P''$, then $P \xrightarrow{\tau} P'$ with $P'' = P'\{y/z\}$, or $P \xrightarrow{(\boldsymbol{\nu}\,\widetilde{b'})\overline{z}\langle\widetilde{b}\rangle} \xrightarrow{y\langle\widetilde{b}\rangle} P'$ with $P'' \equiv (\boldsymbol{\nu}\,\widetilde{b'}\,P')\{y/z\}$, for some P', $\widetilde{b'}$ and $\widetilde{b}$.*
3. *If $P \xrightarrow{\mu} P'$ and $z \notin n(\mu)$, then $P\{y/z\} \xrightarrow{\mu} P'\{y/z\}$.*
4. *If $P \xrightarrow{(\boldsymbol{\nu}\,\widetilde{b'})\overline{z}\langle\widetilde{b}\rangle} P'$ then $P \equiv \boldsymbol{\nu}\,\widetilde{b'}(\overline{z}\langle\widetilde{b}\rangle \mid P')\,.$*

Proof: Tedious but not difficult transition inductions. $\square$

Proposition 6.2 *Suppose that $P \in \mathcal{P}^*$, that y and z are names of the same sort with $y \neq z$, and that z acts as a sender end-point in P. Then*

$$\boldsymbol{\nu}\, z\,(\,!\,z(\widetilde{w}).\,\overline{y}\langle\widetilde{w}\rangle \mid P) \gtrsim P\{y/z\}$$

Proof: Let $\mathcal{R}$ be the set of pairs $(P\{y/z\}, \boldsymbol{\nu}\, z\,(\,!\,z(\widetilde{w}).\,\overline{y}\langle\widetilde{w}\rangle \mid P))$ s.t. P satisfies the conditions on names y, z as in the assertion of the proposition. We prove that $\mathcal{R}$ is an expansion up-to $\equiv$. We first examine the actions of $P\{y/z\}$ and show how $\boldsymbol{\nu}\, z\,(\,!\,z(\widetilde{w}).\,\overline{y}\langle\widetilde{w}\rangle \mid P)$ can match them, and then the converse. In the reasoning below we rely on Lemma 6.1. Suppose

$$P\{y/z\} \xrightarrow{\mu} P''\,. \tag{14}$$

Then one of the following conditions hold:

(a). $P \xrightarrow{\mu'} P'$ and $\mu'\{y/z\} = \mu$, $P'\{y/z\} = P''$;

(b). $\mu = \tau$, $P \xrightarrow{(\boldsymbol{\nu}\,\widetilde{b'})\overline{z}\langle\widetilde{b}\rangle} P_1 \xrightarrow{y\langle\widetilde{b}\rangle} P'$ and $P'' \equiv (\boldsymbol{\nu}\,\widetilde{b'}\,P')\{y/z\}$.

If (a) is true and $z \notin fn(\mu')$, then $\mu = \mu'$ and $\boldsymbol{\nu}\,z\,(\,!\,z(\widetilde{w}).\,\overline{y}\langle\widetilde{w}\rangle|P) \xrightarrow{\mu'} \boldsymbol{\nu}\,z\,(\,!\,z(\widetilde{w}).\,\overline{y}\langle\widetilde{w}\rangle|P')$ with $(P'', \boldsymbol{\nu}\,z\,(\,!\,z(\widetilde{w}).\,\overline{y}\langle\widetilde{w}\rangle \mid P')) \in \mathcal{R}$. If (a) is true and $z \in fn(\mu')$, then for some $\widetilde{b'}$ and $\widetilde{b}$, $\mu' = (\boldsymbol{\nu}\,\widetilde{b'})\overline{z}\langle\widetilde{b}\rangle$ and $\mu = (\boldsymbol{\nu}\,\widetilde{b'})\overline{y}\langle\widetilde{b}\rangle$ and we infer

$$\boldsymbol{\nu}\,z\,(\,!\,z(\widetilde{w}).\,\overline{y}\langle\widetilde{w}\rangle|P) \xrightarrow{\tau}\equiv \boldsymbol{\nu}\,z\,\big(\,!\,z(\widetilde{w}).\,\overline{y}\langle\widetilde{w}\rangle|\boldsymbol{\nu}\,\widetilde{b'}\,(\overline{y}\langle\widetilde{b}\rangle|P')\big) \xrightarrow{(\boldsymbol{\nu}\,\widetilde{b'})\overline{y}\langle\widetilde{b}\rangle}\equiv \boldsymbol{\nu}\,z\,\big(\,!\,z(\widetilde{w}).\,\overline{y}\langle\widetilde{w}\rangle|P'\big)\,.$$

This matches action (14), since $(P'', \boldsymbol{\nu}\,z\,\big(\,!\,z(\widetilde{w}).\,\overline{y}\langle\widetilde{w}\rangle \mid P'\big)) \in \mathcal{R}$ and $\mathcal{R}$ is an expansion up-to $\equiv$. Finally, if (b) is true, then we infer:

$$\boldsymbol{\nu}\,z\,(\,!\,z(\widetilde{w}).\,\overline{y}\langle\widetilde{w}\rangle|P) \xrightarrow{\tau}\equiv \boldsymbol{\nu}\,z\,\big(\,!\,z(\widetilde{w}).\,\overline{y}\langle\widetilde{w}\rangle|\boldsymbol{\nu}\,\widetilde{b'}\,(\overline{y}\langle\widetilde{b}\rangle|P_1)\big) \xrightarrow{\tau}\equiv \boldsymbol{\nu}\,z\,\big(\,!\,z(\widetilde{w}).\,\overline{y}\langle\widetilde{w}\rangle|\boldsymbol{\nu}\,\widetilde{b'}\,P'\big)$$

which, as in the previous case, matches action (14).

Now let us turn to examine the possible moves of $\boldsymbol{\nu}\,z\,(\,!\,z(\widetilde{w}).\,\overline{y}\langle\widetilde{w}\rangle \mid P)$. Since z is restricted, any such action is caused by an action of P, say $P \xrightarrow{\mu} P'$. For the case when $z \notin fn(\mu)$, use Lemma 6.1(3). Suppose $z \in fn(\mu)$. Then for some $\widetilde{b'}$ and $\widetilde{b}$ we have $\mu = (\boldsymbol{\nu}\,\widetilde{b'})\overline{z}\langle\widetilde{b}\rangle$ and $P \equiv \boldsymbol{\nu}\,\widetilde{b'}\,(\overline{z}\langle\widetilde{b}\rangle \mid P')$. From this we infer

$$\boldsymbol{\nu}\,z\,(\,!\,z(\widetilde{w}).\,\overline{y}\langle\widetilde{w}\rangle \mid P) \xrightarrow{\tau}\equiv \boldsymbol{\nu}\,z\,\big(\,!\,z(\widetilde{w}).\,\overline{y}\langle\widetilde{w}\rangle \mid \boldsymbol{\nu}\,\widetilde{b'}\,(\overline{y}\langle\widetilde{b}\rangle \mid P')\big)\,.$$

Process $P\{y/z\}$ does not need to move to match this action, since $P\{y/z\} \equiv \boldsymbol{\nu}\,\widetilde{b'}\,(\overline{y}\langle\widetilde{b}\rangle|P'\{y/z\})$ and processes $\boldsymbol{\nu}\,\widetilde{b'}\,(\overline{y}\langle\widetilde{b}\rangle \mid P'\{y/z\})$ and $\boldsymbol{\nu}\,z\,(\,!\,z(\widetilde{w}).\,\overline{y}\langle\widetilde{w}\rangle \mid \boldsymbol{\nu}\,\widetilde{b'}\,(\overline{y}\langle\widetilde{b}\rangle \mid P')$ are in the relation $\mathcal{R}$. $\square$

We write $P \in \mathcal{L}[\![\mathcal{P}_{loc}]\!]$ if there exists a locating function φ and a located process A s.t. $\big(P, \varphi\big) \in \mathcal{L}[\![A]\!]$.

Corollary 6.3 *Let $P \in \mathcal{L}[\![\mathcal{P}_{loc}]\!]$ and y, z be locating names. Then $\boldsymbol{\nu}\,z\,(\,!\,z.\,\overline{y}|P) \gtrsim P\{y/z\}$.*

PROOF: Name z only occurs free in $P \in \mathcal{L}[\![\mathcal{P}_{loc}]\!]$ in subexpressions of the form $\mathrm{Loc}\langle xz'z\rangle$. If we expand the definition of Loc, replacing $\mathrm{Loc}\langle xz'z\rangle$ with $\overline{x}\langle z'\rangle.\ \mid\ !\,z'.\,\overline{z}$, process P becomes a process of $\mathcal{P}^*$ in which z acts as a sender end-point. Then the assertion of the corollary follows from Proposition 6.2. $\square$

Corollary 6.3 is one of the key results for the proof of the correctness of $\mathcal{L}$. A consequence of this corollary which we shall use is the following.

Corollary 6.4 *If P_1 and $P_2 \in \mathcal{L}[\![\mathcal{P}_{loc}]\!]$, then*

$$\boldsymbol{\nu}\, x\, \big(\boldsymbol{\nu}\, z\, (\mathrm{Loc}\langle xzy_1\rangle \mid P_1) \mid \boldsymbol{\nu}\, z\, (\mathrm{Loc}\langle xzy_2\rangle \mid P_2)\big) \gtrsim P_1\{y_1/z\} \mid P_2\{y_2/z\}\,.$$

PROOF: Let

$$P \stackrel{def}{=} \boldsymbol{\nu}\, z\, (\mathrm{Loc}\langle xzy_1\rangle \mid P_1) \mid \boldsymbol{\nu}\, z\, (\mathrm{Loc}\langle xzy_2\rangle \mid P_2)\,.$$

We want to prove that $\boldsymbol{\nu}\, x\, P \gtrsim P_1\{y_1/z\} \mid P_2\{y_2/z\}$. By definition of the constant Loc, we have

$$\mathrm{Loc}\langle xzy_1\rangle = \overline{x}\langle z\rangle \mid !\, z.\, \overline{y_1} \qquad \text{and} \qquad \mathrm{Loc}\langle xzy_2\rangle = \overline{x}\langle z\rangle \mid !\, z.\, \overline{y_2}.$$

The particles $\overline{x}\langle z\rangle$ contain the only free occurrences of name x in P; but since x is restricted in $\boldsymbol{\nu}\, x\, P$, such particles are deadlocked, and can be canceled. Therefore,

$$\boldsymbol{\nu}\, x\, P \gtrsim \boldsymbol{\nu}\, z\, (\,!\, z.\, \overline{y_1} \mid P_1) \mid \boldsymbol{\nu}\, z\, (\,!\, z.\, \overline{y_2} \mid P_2)\,.$$

From here, the assertion of the lemma follows applying Corollary 6.3 and the congruence of $\gtrsim$ w.r.t. parallel composition twice. $\square$

7 Operational correspondence for $\mathcal{L}$

In this section we show that if $(P, \varphi) \in \mathcal{L}[\![A]\!]$, then there is a precise operational correspondence between the actions performed by A and P. We consider strong transitions in Subsection 7.1 and we extend the results to weak transitions in Subsection 7.2. First, we need a few notations and facts about locating functions. For a locating function $\varphi = \{(y_1, u_1), \ldots, (y_n, u_n)\}$, we refer to $\{y_1, \ldots, y_n\}$ as $\mathtt{dom}(\varphi)$ and to $\{u_1, \ldots, u_n\}$ as $\mathtt{cod}(\varphi)$. If $y \notin \mathtt{dom}(\varphi)$ and $u \notin \mathtt{cod}(\varphi)$, then we write $\varphi \cup (y, u)$ for the locating function which extends φ with the pair (y, u). We write $\varphi \subseteq \varphi'$ if $(y, u) \in \varphi$ implies $(y, u) \in \varphi'$. Some simple properties which follow from the definitions $\mathcal{S}$ and $\mathcal{L}$ are:

Lemma 7.1

1. *If b and c are not locating names and $(P, \varphi) \in \mathcal{L}[\![A]\!]$, then also $(P\{b/c\}, \varphi) \in \mathcal{L}[\![A\{b/c\}]\!]$.*
2. *If $(P, \varphi) \in \mathcal{L}[\![A]\!]$ and z is a locating name not in $\mathtt{dom}(\varphi)$, then $z \notin fn(P)$.*
3. *If φ and φ' are two locating functions with $\varphi \subseteq \varphi'$, then $(P, \varphi) \in \mathcal{L}[\![A]\!]$ implies $(P, \varphi') \in \mathcal{L}[\![A]\!]$.*
4. *If $(P, \varphi) \in \mathcal{L}[\![A]\!]$, then the set of free names of A coincides with the set of free names of P which are not locating names.*

7.1 Operational correspondence for $\mathcal{L}$ on strong transitions

Lemma 7.2 (operational correspondence for $\mathcal{L}$ on strong location transitions)

Let $\big(P, \varphi\big) \in \mathcal{L}[\![A]\!]$, *and* $z \notin \mathtt{dom}(\varphi)$*:*

1. (a) *if* $A \xrightarrow[ul]{a\langle\widetilde{b}\rangle} A'$, *with* $ul \notin \mathtt{cod}(\varphi)$, *then there are* P' *and* y *s.t. for all* x,

$$P \xrightarrow{a\langle\widetilde{bx}\rangle} \equiv \boldsymbol{\nu}\, z\,(\mathrm{Loc}\langle xzy\rangle \mid P')$$

with $y = \varphi^{-1}(u)$ *and* $\big(P', \varphi \cup (z, ul)\big) \in \mathcal{L}[\![A']\!]$.

   (b) *The converse, i.e. if* $P \xrightarrow{a\langle\widetilde{bx}\rangle} P''$, *then there are* A', P', y *and* u, *s.t. for all* l *with* $ul \notin \mathtt{cod}(\varphi)$ *we have*

$$A \xrightarrow[ul]{a\langle\widetilde{b}\rangle} A' \qquad \textit{and} \qquad P'' \equiv \boldsymbol{\nu}\, z\,(\mathrm{Loc}\langle xzy\rangle \mid P')$$

with $y = \varphi^{-1}(u)$ *and* $\big(P', \varphi \cup (z, ul)\big) \in \mathcal{L}[\![A']\!]$.

2. (a) *If* $A \xrightarrow[ul]{(\boldsymbol{\nu}\,\widetilde{b'})\overline{a}\langle\widetilde{b}\rangle} A'$, *with* $ul \notin \mathtt{cod}(\varphi)$, *then there are* P' *and* y *s.t. for all* x,

$$P \xrightarrow{(\boldsymbol{\nu}\,\widetilde{b'x})\overline{a}\langle\widetilde{bx}\rangle} \equiv \boldsymbol{\nu}\, z\,(\mathrm{Loc}\langle xzy\rangle \mid P')$$

with $y = \varphi^{-1}(u)$ *and* $\big(P', \varphi \cup (z, ul)\big) \in \mathcal{L}[\![A']\!]$.

   (b) *The converse, i.e. if* $P \xrightarrow{(\boldsymbol{\nu}\,\widetilde{b'x})\overline{a}\langle\widetilde{bx}\rangle} P''$, *then there are* A', P', y *and* u, *s.t. for all* l *with* $ul \notin \mathtt{cod}(\varphi)$ *we have*

$$A \xrightarrow[ul]{(\boldsymbol{\nu}\,\widetilde{b'})\overline{a}\langle\widetilde{b}\rangle} A', \qquad \textit{and} \qquad P'' \equiv \boldsymbol{\nu}\, z\,(\mathrm{Loc}\langle xzy\rangle \mid P')$$

with $y = \varphi^{-1}(u)$ *and* $\big(P', \varphi \cup (z, ul)\big) \in \mathcal{L}[\![A']\!]$.

Proof: We only consider clause (1) since (2) can be handled in a similar way. First we look at (1a), and proceed by transition induction. The basis of the induction occurs when the rule used is `L-INP`. Then we have $A = a(\widetilde{c}).\, Q$, $u = \emptyset$ and $P = \mathcal{S}[\![a(\widetilde{c}).\, Q]\!]\langle y\rangle$, for $y = \varphi^{-1}(\emptyset)$. The actions that A and P can perform are:

$$A \xrightarrow[l]{a\langle\widetilde{b}\rangle} l :: Q\{\widetilde{b}/\widetilde{c}\}, \text{ for any } l$$

and

$$P \xrightarrow{a\langle\widetilde{bx}\rangle} \boldsymbol{\nu}\, z\,(\mathrm{Loc}\langle xzy\rangle \mid \mathcal{S}[\![Q]\!]\langle z\rangle\{\widetilde{b}/\widetilde{c}\}) = \boldsymbol{\nu}\, z\,(\mathrm{Loc}\langle xzy\rangle \mid \mathcal{S}[\![Q\{\widetilde{b}/\widetilde{c}\}]\!]\langle z\rangle), \text{ for any } x.$$

Assuming $l \notin \mathtt{cod}(\varphi)$ and, by alpha conversion, that $z \notin \mathtt{dom}(\varphi)$, we have:

$$\big(\mathcal{S}[\![Q\{\tilde{b}/\tilde{c}\}]\!]\langle z\rangle, \varphi \cup (z,l)\big) \in \mathcal{L}[\![l :: Q\{\tilde{b}/\tilde{c}\}]\!]\,.$$

Now we look at the inductive part. We proceed by a case analysis on the last rule used to infer $A \xrightarrow[ul]{a\langle\tilde{b}\rangle} A'$.

Case (a) Rule L-LOC.

We have $A = u :: B$ and the last inference is of the form

$$\frac{B \xrightarrow[vl]{a\langle\tilde{b}\rangle} B'}{u :: B \xrightarrow[uvl]{a\langle\tilde{b}\rangle} u :: B'}$$

By definition of $\mathcal{L}$, $\big(P,\varphi\big) \in \mathcal{L}[\![u :: B]\!]$ if $(P,\varphi_{-u}) \in \mathcal{L}[\![B]\!]$. From the inductive assumption,

$$P \xrightarrow{a\langle\tilde{b}x\rangle} \equiv \boldsymbol{\nu}\, z\,(\mathrm{Loc}\langle xzy\rangle \mid P')$$

with $y = \varphi_{-u}^{-1}(v)$ and $\big(P', \varphi_{-u} \cup (z,vl)\big) \in \mathcal{L}[\![B']\!]$. Therefore, we also have $y = \varphi^{-1}(uv)$ and $(P_2, \varphi \cup (z,uvl)) \in \mathcal{L}[\![u :: B']\!]$, which concludes the case.

Case (b) Rule L-PAR.

We have $A = A_1 \mid A_2$ and $P = P_1 \mid P_2$ with $\big(P_i,\varphi\big) \in \mathcal{L}[\![A_i]\!]$, $i = 1,2$. Assuming that action $A \xrightarrow[ul]{a\langle\tilde{b}\rangle} A'$ has been caused by A_1, we have $A_1 \xrightarrow[ul]{a\langle\tilde{b}\rangle} A_1'$ and $A' = A_1' \mid A_2$. Then, by induction,

$$P_1 \xrightarrow{a\langle\tilde{b}x\rangle} \equiv \boldsymbol{\nu}\, z\,(\mathrm{Loc}\langle xzy\rangle \mid P_1')$$

with $y = \varphi^{-1}(u)$ and

$$\big(P_1', \varphi \cup (z,ul)\big) \in \mathcal{L}[\![A_1']\!]\,. \tag{15}$$

Since $z \notin \mathtt{dom}(\varphi)$, by Lemma 7.1(2) also $z \notin fn(P_2)$ and therefore

$$P_1 \mid P_2 \xrightarrow{a\langle\tilde{b}x\rangle} \equiv \boldsymbol{\nu}\, z\,(\mathrm{Loc}\langle xzy\rangle \mid (P_1' \mid P_2))\,.$$

Moreover, by $\big(P_2,\varphi\big) \in \mathcal{L}[\![A_2]\!]$ and Lemma 7.1(3) we get $\big(P_2, \varphi \cup (z,ul)\big) \in \mathcal{L}[\![A_2]\!]$. From this and (15) we get $\big(P_1' \mid P_2, \varphi \cup (z,ul)\big) \in \mathcal{L}[\![A_1' \mid A_2]\!]$.

Case (c) Rule L-RES, L-MATCH or L-SUM.

Use the inductive assumption and simple structural congruence rules.

Case (d) Rule `L-CONS`.

We have $A = D\langle\widetilde{b}\rangle$ and $P = D_E\langle\widetilde{b}y\rangle$ with $y = \varphi^{-1}(\emptyset\,)$; we are using different denominations for D and its encoding D_E for readability purposes. Using Lemma 4.6, the last inference is of the form

$$\frac{Q\{\widetilde{b}/\widetilde{c}\} \xrightarrow[l]{a\langle\widetilde{b}\rangle} A'}{D\langle\widetilde{b}\rangle \xrightarrow[l]{a\langle\widetilde{b}\rangle} A'}\; D \stackrel{def}{=} (\widetilde{c})Q$$

We have $D_E \stackrel{def}{=} (\widetilde{c}y)\mathcal{S}[\![Q]\!]\langle y\rangle$. Moreover, by induction,

$$\mathcal{S}[\![Q]\!]\langle y\rangle\{\widetilde{b}/\widetilde{c}\} = \mathcal{S}[\![Q\{\widetilde{b}/\widetilde{c}\}]\!]\langle y\rangle \xrightarrow{a\langle\widetilde{b}x\rangle} \equiv \boldsymbol{\nu}\, z\,(\mathrm{Loc}\langle xzy\rangle \mid Q') \tag{16}$$

with

$$y = \varphi^{-1}(\emptyset\,) \quad \text{and} \quad \big(Q', \varphi \cup (z, l)\big) \in \mathcal{L}[\![A']\!]\,. \tag{17}$$

Therefore, from (16) and rule `S-CONS` we infer

$$D_E\langle\widetilde{b}y\rangle \xrightarrow{a\langle\widetilde{b}x\rangle} \equiv \boldsymbol{\nu}\, z\,(\mathrm{Loc}\langle xzy\rangle \mid Q')$$

which, together with (17), concludes the case.

For clause (1b), first show that the assertion holds if A is a standard process, using transition induction. Then extend the result to located processes, using structural induction on A. $\square$

Now, we wish to study the correspondence for $\mathcal{L}$ on silent transitions. For this, we first have to analyse the correspondence on visible standard transitions.

Lemma 7.3 (operational correspondence for $\mathcal{L}$ on visible standard transitions)

Let $\big(P, \varphi\big) \in \mathcal{L}[\![A]\!]$.

1. *(a) If* $A \xrightarrow{a\langle\widetilde{b}\rangle} A'$*, then there are* $\widetilde{d}$*,* A_1*,* A_2*,* P_1*,* P_2 *and* y*, s.t.*

$$A' \equiv \boldsymbol{\nu}\,\widetilde{d}\,(A_1 \mid A_2) \quad \textit{and} \quad P \xrightarrow{a\langle\widetilde{b}x\rangle} \equiv \boldsymbol{\nu}\,\widetilde{d}\,\big(\boldsymbol{\nu}\, z\,(\mathrm{Loc}\langle xzy\rangle \mid P_1) \mid P_2\big)$$

with $\big(P_i\{y/z\}, \varphi\big) \in \mathcal{L}[\![A_i]\!]$*,* $i = 1, 2$.

(b) The converse, i.e. if $P \xrightarrow{a\langle\widetilde{b}x\rangle} P''$ *then for some* $\widetilde{d}, A_1$*,* A_2*,* P_1*,* P_2 *and* y *we have*

$$A \xrightarrow{a\langle\widetilde{b}\rangle} \equiv \boldsymbol{\nu}\,\widetilde{d}\,(A_1 \mid A_2) \quad \textit{and} \quad P'' \equiv \boldsymbol{\nu}\,\widetilde{d}\,\big(\boldsymbol{\nu}\, z\,(\mathrm{Loc}\langle xzy\rangle \mid P_1) \mid P_2\big)$$

with $\big(P_i\{y/z\}, \varphi\big) \in \mathcal{L}[\![A_i]\!]$, $i = 1, 2$.

2. (a) *If* $A \overset{(\boldsymbol{\nu}\,\widetilde{b'})\overline{a}\langle\widetilde{b}\rangle}{\longrightarrow} A'$, *then there are* $\widetilde{d}$, A_1, A_2, P_1, P_2 *and* y, *s.t.*

$$A' \equiv \boldsymbol{\nu}\,\widetilde{d}\,(A_1 \mid A_2) \qquad and \qquad P \overset{(\boldsymbol{\nu}\,\widetilde{b'x})\overline{a}\langle\widetilde{bx}\rangle}{\longrightarrow} \equiv \boldsymbol{\nu}\,\widetilde{d}\big(\boldsymbol{\nu}\,z\,(\mathrm{Loc}\langle xzy\rangle \mid P_1) \mid P_2\big)$$

with $\big(P_i\{y/z\}, \varphi\big) \in \mathcal{L}[\![A_i]\!]$, $i = 1, 2$.

(b) *The converse, i.e. if* $P \overset{(\boldsymbol{\nu}\,\widetilde{b'x})\overline{a}\langle\widetilde{bx}\rangle}{\longrightarrow} P''$ *then for some* $\widetilde{d}$, A_1, A_2, P_1, P_2 *and* y *we have*

$$A \overset{(\boldsymbol{\nu}\,\widetilde{b'})a\langle\widetilde{b}\rangle}{\longrightarrow} \equiv \boldsymbol{\nu}\,\widetilde{d}\,(A_1 \mid A_2) \qquad and \qquad P'' \equiv \boldsymbol{\nu}\,\widetilde{d}\big(\boldsymbol{\nu}\,z\,(\mathrm{Loc}\langle xzy\rangle \mid P_1) \mid P_2\big)$$

with $\big(P_i\{y/z\}, \varphi\big) \in \mathcal{L}[\![A_i]\!]$, $i = 1, 2$.

Proof: We only consider clause (1a), and proceed by transition induction. The basis of the induction occurs when the rule used is L-INP. Then $A = a(\widetilde{c}).\,Q$ and $P = \mathcal{S}[\![a(\widetilde{c}).\,Q]\!]\langle y\rangle$ with $y = \varphi^{-1}(\emptyset\,)$. For this, take $\widetilde{d} = \emptyset$, $A_2 \overset{def}{=} \mathbf{0}$, and $A_1 \overset{def}{=} Q\{\widetilde{b}/\widetilde{c}\}$; then everything follows easily. Now we look at the inductive part. We proceed by a case analysis on the last rule used to infer $A \overset{a\langle\widetilde{b}\rangle}{\longrightarrow} A'$.

Case (a) Rule L-LOC.

Then $A = u :: B$ and the last inference is of the form

$$\frac{B \overset{a\langle\widetilde{b}\rangle}{\longrightarrow} B'}{u :: B \overset{a\langle\widetilde{b}\rangle}{\longrightarrow} u :: B'}$$

By definition of $\mathcal{L}$, $\big(P, \varphi\big) \in \mathcal{L}[\![u :: B]\!]$ if $(P, \varphi_{-u}) \in \mathcal{L}[\![B]\!]$. From the inductive assumption, $B' \equiv \boldsymbol{\nu}\,\widetilde{d}\,(B_1 \mid B_2)$ and

$$P \overset{a\langle\widetilde{bx}\rangle}{\longrightarrow} \equiv \boldsymbol{\nu}\,\widetilde{d}\big(\boldsymbol{\nu}\,z\,(\mathrm{Loc}\langle xzy\rangle \mid P_1') \mid P_2'\big)$$

with $\big(P_i'\{y/z\}, \varphi_{-u}\big) \in \mathcal{L}[\![B_i]\!]$, $i = 1, 2$. Therefore, we also have $u :: B' \equiv \boldsymbol{\nu}\,\widetilde{d}\,(u :: B_1 \mid u :: B_2)$ and $\big(P_i'\{y/z\}, \varphi\big) \in \mathcal{L}[\![u :: B_i]\!]$, $i = 1, 2$.

Case (b) Rule L-PAR.

We have $A = A_1 \mid A_2$ and $P = P_1 \mid P_2$ with

$$\big(P_i, \varphi\big) \in \mathcal{L}[\![A_i]\!], \quad i = 1, 2. \tag{18}$$

Assuming that action $A \xrightarrow{a\langle\widetilde{b}\rangle} A'$ has been caused by A_1, we have $A_1 \xrightarrow{a\langle\widetilde{b}\rangle} A_1'$ and $A' = A_1' \mid A_2$. Then, by induction, $A_1' \equiv \boldsymbol{\nu}\, \widetilde{d}\,(B_1 \mid B_2)$ and, assuming $\widetilde{d} \cap fn(A_2) = \emptyset$, also

$$A' \equiv \boldsymbol{\nu}\, d\,(B_1 \mid (B_2 \mid A_2)).$$

Moreover

$$P_1 \xrightarrow{a\langle\widetilde{b}x\rangle} \equiv \boldsymbol{\nu}\, \widetilde{d}\Big(\boldsymbol{\nu}\, z\,(\mathrm{Loc}\langle xzy\rangle \mid Q_1) \mid Q_2\Big)$$

with

$$\Big(Q_i\{y/z\}, \varphi\Big) \in \mathcal{L}[\![B_i]\!], \quad i = 1, 2. \tag{19}$$

Hence, assuming $\widetilde{d} \cap fn(P_2) = \emptyset$, we infer

$$P_1 \mid P_2 \xrightarrow{a\langle\widetilde{b}x\rangle} \equiv \boldsymbol{\nu}\, \widetilde{d}\Big(\boldsymbol{\nu}\, z\,(\mathrm{Loc}\langle xzy\rangle \mid Q_1) \mid (Q_2 \mid P_2)\Big).$$

Moreover it holds:

- From (19), $\Big(Q_1\{y/z\}, \varphi\Big) \in \mathcal{L}[\![B_1]\!]$;
- from (18) and (19) and definition of $\mathcal{L}$, $\Big((Q_2 \mid P_2)\{y/z\}, \varphi\Big) \in \mathcal{L}[\![B_2 \mid A_2]\!]$, for by alpha conversion we can assume that z is not free in P_2.

Case (c) Rule L-RES, L-MATCH or L-SUM.

Follows easily using the inductive assumption.

Case (d) Rule L-CONS.

We have $A = D\langle\widetilde{b}\rangle$ and, calling D_E the translation of D, we have $P = D_E\langle\widetilde{b}y\rangle$ with $y = \varphi^{-1}(\emptyset\,)$. The last inference is of the form

$$\frac{Q\{\widetilde{b}/\widetilde{c}\} \xrightarrow{a\langle\widetilde{b}\rangle} A'}{D\langle\widetilde{b}\rangle \xrightarrow{a\langle\widetilde{b}\rangle} A'}\; D \stackrel{def}{=} (\widetilde{c})Q\,.$$

We have $D_E \stackrel{def}{=} (\widetilde{c}y)\mathcal{S}[\![Q]\!]\langle y\rangle$. Moreover, by induction, $A' \equiv \boldsymbol{\nu}\, \widetilde{d}\,(A_1' \mid A_2')$ and

$$\mathcal{S}[\![Q]\!]\langle y\rangle\{\widetilde{b}/\widetilde{c}\} = \mathcal{S}[\![Q\{\widetilde{b}/\widetilde{c}\}]\!]\langle y\rangle \xrightarrow{a\langle\widetilde{b}x\rangle} \equiv \boldsymbol{\nu}\, \widetilde{d}\Big(\boldsymbol{\nu}\, z\,(\mathrm{Loc}\langle xzy\rangle \mid Q_1) \mid Q_2\Big) \tag{20}$$

with

$$\Big(Q_i\{y/z\}, \varphi\Big) \in \mathcal{L}[\![A_i']\!], \;\; i = 1, 2. \tag{21}$$

Using (20) and rule S-CONS, we infer $D_E\langle\widetilde{b}y\rangle \xrightarrow{a\langle\widetilde{b}x\rangle} \equiv \boldsymbol{\nu}\, \widetilde{d}\Big(\boldsymbol{\nu}\, z\,(\mathrm{Loc}\langle xzy\rangle \mid Q_1) \mid Q_2\Big)$ which, together with (21), concludes the case.

$\square$

Lemma 7.4 (operational correspondence for $\mathcal{L}$ on strong silent transitions) *Let $(P, \varphi) \in \mathcal{L}[\![A]\!]$.*

1. *If $A \longrightarrow A''$, then there are P' and $A' \equiv A''$ s.t. $P \longrightarrow\gtrsim P'$ with $(P', \varphi) \in \mathcal{L}[\![A']\!]$;*
2. *The converse, i.e. if $P \longrightarrow P''$, then there are A' and $P' \lesssim P''$ s.t. $A \longrightarrow\equiv A'$ and $(P', \varphi) \in \mathcal{L}[\![A']\!]$.*

Proof: We only consider clause (1), as (2) follows with similar reasoning, and proceed by transition induction. The interesting case is rule `L-COM`. Then $A = A_1 \,|\, A_2$ and $P = P_1 \,|\, P_2$ with $(P_i, \varphi) \in \mathcal{L}[\![A_i]\!]$, $i = 1, 2$. Assuming A_1 is the process performing the input, we have

$$A_1 \xrightarrow{a\langle \widetilde{b} \rangle} A_1', \quad A_2 \xrightarrow{(\boldsymbol{\nu}\, \widetilde{b'})\overline{a}\langle \widetilde{b} \rangle} A_2' \qquad \text{and} \qquad A'' = \boldsymbol{\nu}\, \widetilde{b'}\, (A_1' \mid A_2') \,.$$

By Lemma 7.3,

$$A_1' \equiv \boldsymbol{\nu}\, \widetilde{d}\, (B_1 \mid B_2), \quad A_2' \equiv \boldsymbol{\nu}\, \widetilde{e}\, (C_1 \mid C_2)$$

and

$$\begin{array}{lcl} P_1 & \xrightarrow{a\langle \widetilde{b}x \rangle}\equiv & \boldsymbol{\nu}\, \widetilde{d}\, \big(\boldsymbol{\nu}\, z\, (\mathrm{Loc}\langle xzy \rangle \mid Q_1) \mid Q_2\big) \\ P_2 & \xrightarrow{(\boldsymbol{\nu}\, \widetilde{b'}x\,)\overline{a}\langle \widetilde{b}x \rangle}\equiv & \boldsymbol{\nu}\, \widetilde{e}\, \big(\boldsymbol{\nu}\, z\, (\mathrm{Loc}\langle xzy' \rangle \mid R_1) \mid R_2\big) \end{array}$$

with

$$\big(Q_i\{y/z\}, \varphi\big) \in \mathcal{L}[\![B_i]\!], \quad \big(R_i\{y'/z\}, \varphi\big) \in \mathcal{L}[\![C_i]\!] \,. \tag{22}$$

Further, since $z \notin fn(Q_2, R_2)$, also

$$\big(Q_2, \varphi\big) \in \mathcal{L}[\![B_2]\!], \quad \big(R_2, \varphi\big) \in \mathcal{L}[\![C_2]\!] \,. \tag{23}$$

From what obtained above and alpha conversion, we deduce that

$$A'' \equiv \boldsymbol{\nu}\, \widetilde{b'}\widetilde{d}\widetilde{e}\, \big(B_1 \mid C_1 \mid B_2 \mid C_2\big) \stackrel{def}{=} A'$$

and

$$P_1 \mid P_2 \longrightarrow\equiv \boldsymbol{\nu}\, \widetilde{b'}\widetilde{d}\widetilde{e}\, \Big(\boldsymbol{\nu}\, x\, \big(\boldsymbol{\nu}\, z\, (\mathrm{Loc}\langle xzy \rangle \mid Q_1) \mid \boldsymbol{\nu}\, z\, (\mathrm{Loc}\langle xzy' \rangle \mid R_1)\big) \mid Q_2 \mid R_2\Big) \stackrel{def}{=} P'' \,.$$

By Corollary 6.4, $\boldsymbol{\nu}\, x\, \big(\boldsymbol{\nu}\, z\, (\mathrm{Loc}\langle xzy \rangle \mid Q_1) \mid \boldsymbol{\nu}\, z\, (\mathrm{Loc}\langle xzy' \rangle \mid R_1)\big) \gtrsim Q_1\{y/z\} \mid R_1\{y'/z\}$ and, therefore, using the congruence properties of $\gtrsim$,

$$P'' \gtrsim \boldsymbol{\nu}\, \widetilde{b'}\widetilde{d}\widetilde{e}\, \big(Q_1\{y/z\} \mid R_1\{y'/z\} \mid Q_2 \mid R_2\big) \stackrel{def}{=} P' \,.$$

Finally, (22), (23), and the definitions of A' and P', yield $(P', \varphi) \in \mathcal{L}[\![A']\!]$. $\square$

7.2 Operational correspondence on weak transitions

The results in the previous sections are exploited in this one to give the operational correspondence for $\mathcal{L}$ on weak transitions. First, we analyse silent actions, obtaining the analogous of Lemma 7.4.

Lemma 7.5 (operational correspondence for $\mathcal{L}$ on weak silent transitions) *Let $\big(P, \varphi\big) \in \mathcal{L}[\![A]\!]$.*

1. *If $A \Longrightarrow A''$, then there are P' and $A' \equiv A''$ s.t.*
$$P \Longrightarrow\gtrsim P' \quad \textit{and} \quad \big(P', \varphi\big) \in \mathcal{L}[\![A']\!] \,.$$

2. *The converse, i.e. if $P \Longrightarrow P''$, then there are A' and $P' \lesssim P''$ s.t.*
$$A \Longrightarrow\equiv A' \quad \textit{and} \quad \big(P', \varphi\big) \in \mathcal{L}[\![A']\!] \,.$$

Proof: We first consider assertion (1). If $A \Longrightarrow A''$, then for some $m \geq 0$, it holds that $A \longrightarrow^m A''$. We prove the lemma by induction on m. If $m = 0$ there is nothing to prove. Suppose $m > 0$. Then $A \longrightarrow^{m-1} B \longrightarrow A''$. From the inductive assumption, for some Q and $B' \equiv B$,
$$P \Longrightarrow P'' \gtrsim Q \quad \text{and} \quad \big(Q, \varphi\big) \in \mathcal{L}[\![B']\!] \,.$$
Since $B' \equiv B \longrightarrow A''$, by Lemma 4.5 there is B'' s.t. $B' \longrightarrow B'' \equiv A''$. Since $\big(Q, \varphi\big) \in \mathcal{L}[\![B']\!]$ and $B' \longrightarrow B''$, by Lemma 7.4, there are P' and $A' \equiv B''$ s.t. $Q \longrightarrow\gtrsim P'$ and $\big(P', \varphi\big) \in \mathcal{L}[\![A']\!]$. Moreover, since $P'' \gtrsim Q$ and $Q \longrightarrow\gtrsim P'$, for some P''', it holds that $P'' \Longrightarrow P''' \gtrsim P'$. Summarising, we have derived that
$$A \Longrightarrow A'' \equiv B'' \equiv A'$$
and
$$P \Longrightarrow P'' \Longrightarrow P''' \gtrsim P'$$
with $\big(P', \varphi\big) \in \mathcal{L}[\![A']\!]$, which concludes the case.

Now, we examine (2). Again, we proceed by induction on the m s.t. $P \longrightarrow^m P''$. In the basic case, $m = 0$ and there is nothing to prove. Suppose $m > 0$. Then
$$P \longrightarrow^{m-1} Q \longrightarrow P'' \,.$$
By induction, for some $Q' \lesssim Q$ and B, B':
$$A \Longrightarrow B \equiv B' \quad \text{and} \quad \big(Q', \varphi\big) \in \mathcal{L}[\![B']\!] \,.$$

Since $Q' \lesssim Q$ and $Q \longrightarrow P''$, by definition of $\lesssim$, we have $Q' \stackrel{\wedge}{\longrightarrow} P''' \lesssim P''$. From this and $\big(Q', \varphi\big) \in \mathcal{L}[\![B']\!]$, using Lemma 7.4 we have, for some $P' \lesssim P'''$ and A',

$$B' \stackrel{\wedge}{\longrightarrow}\equiv A' \qquad \text{and} \qquad \big(P', \varphi\big) \in \mathcal{L}[\![A']\!]\,.$$

Moreover, from $B' \equiv B$ and Lemma 4.5, also $B \stackrel{\wedge}{\longrightarrow}\equiv A'$. Summarising, we have derived that

$$P \longrightarrow^{m-1} Q \longrightarrow P'' \gtrsim P''' \gtrsim P' \qquad \text{and} \qquad A \Longrightarrow B \stackrel{\wedge}{\longrightarrow}\equiv A'$$

with $\big(P', \varphi\big) \in \mathcal{L}[\![A']\!]$. This concludes the proof. $\square$

The use of the expansion relation was *necessary* to get the proof of Lemma 7.5(2): Had we used $\approx$ instead of $\gtrsim$, we could have not closed the inductive case of the proof. Next, we consider weak visible transitions, and we derive the analogous of Lemma 7.2.

Lemma 7.6 (operational correspondence for $\mathcal{L}$ on weak location transitions) *Let $\big(P, \varphi\big) \in \mathcal{L}[\![A]\!]$ and $z \notin \mathtt{dom}(\varphi)$.*

1. *(a) If $A \overset{a\langle\widetilde{b}\rangle}{\underset{ul}{\Longrightarrow}} A''$ with $ul \notin \mathtt{cod}(\varphi)$, then there are P', y and $A' \equiv A''$, s.t. for all x,*

$$P \overset{a\langle\widetilde{bx}\rangle}{\Longrightarrow} \gtrsim \boldsymbol{\nu}\, z\,(\mathrm{Loc}\langle xzy\rangle \mid P')$$

   with $y = \varphi^{-1}(u)$ and $\big(P', \varphi \cup (z, ul)\big) \in \mathcal{L}[\![A']\!]$.

   (b) The converse, i.e. if $P \overset{a\langle\widetilde{bx}\rangle}{\Longrightarrow} P''$, then there are A', P', y and u s.t. for all l with $ul \notin \mathtt{cod}(\varphi)$, we have

$$A \overset{a\langle\widetilde{b}\rangle}{\underset{ul}{\Longrightarrow}} \equiv A' \qquad \text{and} \qquad P'' \gtrsim \boldsymbol{\nu}\, z\,(\mathrm{Loc}\langle xzy\rangle \mid P')$$

   with $y = \varphi^{-1}(u)$ and $\big(P', \varphi \cup (z, ul)\big) \in \mathcal{L}[\![A']\!]$.

2. *(a) If $A \overset{(\boldsymbol{\nu}\,\widetilde{b'})\overline{a}\langle\widetilde{b}\rangle}{\underset{ul}{\Longrightarrow}} A''$ with $ul \notin \mathtt{cod}(\varphi)$, then there are P', y and $A' \equiv A''$, s.t. for all x,*

$$P \overset{(\boldsymbol{\nu}\,\widetilde{b'x})\overline{a}\langle\widetilde{bx}\rangle}{\Longrightarrow} \gtrsim \boldsymbol{\nu}\, z\,(\mathrm{Loc}\langle xzy\rangle \mid P')$$

   with $y = \varphi^{-1}(u)$ and $\big(P', \varphi \cup (z, ul)\big) \in \mathcal{L}[\![A']\!]$.

   (b) The converse, i.e. if $P \overset{(\boldsymbol{\nu}\,\widetilde{b'x})\overline{a}\langle\widetilde{bx}\rangle}{\Longrightarrow} P''$, then there are A', P', y and u s.t. for all l with $ul \notin \mathtt{cod}(\varphi)$, we have

$$A \overset{(\boldsymbol{\nu}\,\widetilde{b'})\overline{a}\langle\widetilde{b}\rangle}{\underset{ul}{\Longrightarrow}} \equiv A' \qquad \text{and} \qquad P'' \gtrsim \boldsymbol{\nu}\, z\,(\mathrm{Loc}\langle xzy\rangle \mid P')$$

   with $y = \varphi^{-1}(u)$ and $\big(P', \varphi \cup (z, ul)\big) \in \mathcal{L}[\![A']\!]$.

PROOF: We only consider clause (1a) in detail. If $A \overset{a\langle\widetilde{b}\rangle}{\underset{ul}{\Longrightarrow}} A''$, then there are B' and B'' s.t.

$$A \Longrightarrow B' \overset{a\langle\widetilde{b}\rangle}{\underset{ul}{\longrightarrow}} B'' \Longrightarrow A''. \tag{24}$$

Since $\big(P,\varphi\big) \in \mathcal{L}[\![A]\!]$, by Lemma 7.5, there are B_1 and Q_1, s.t. $B' \equiv B_1$ and

$$P \Longrightarrow\gtrsim Q_1 \quad \text{ with } \quad \big(Q_1,\varphi\big) \in \mathcal{L}[\![B_1]\!]. \tag{25}$$

Since $B' \overset{a\langle\widetilde{b}\rangle}{\underset{ul}{\longrightarrow}} B''$ and $B_1 \equiv B$, by Lemma 4.5

$$B_1 \overset{a\langle\widetilde{b}\rangle}{\underset{ul}{\longrightarrow}} B_2 \equiv B''. \tag{26}$$

Hence, from (25), (26) and Lemma 7.2, there are Q_2 and y s.t. for all x,

$$Q_1 \overset{a\langle\widetilde{b}x\rangle}{\longrightarrow} \equiv \boldsymbol{\nu}\, z\,(\mathrm{Loc}\langle xzy\rangle \mid Q_2) \tag{27}$$

with

$$y = \varphi^{-1}(u) \quad \text{ and } \quad \big(Q_2, \varphi \cup (z, ul)\big) \in \mathcal{L}[\![B_2]\!]. \tag{28}$$

Finally, since $B'' \Longrightarrow A''$, from (26) and Lemma 4.5, we get

$$B_2 \Longrightarrow B_3 \equiv A'' \tag{29}$$

and then, from (28) and Lemma 7.5(1), for some B_4 and Q_3, we have

$$B_3 \equiv B_4 \tag{30}$$

$$Q_2 \Longrightarrow\gtrsim Q_3 \quad \text{ with } \quad \big(Q_3, \varphi \cup (z, ul)\big) \in \mathcal{L}[\![B_4]\!]. \tag{31}$$

Summarising, we have:

- by (29) and (30), $A'' \equiv B_3 \equiv B_4$,
- using (25), (27) and (31), and the congruence properties of $\lesssim$, we can infer

  $$P \Longrightarrow \overset{a\langle\widetilde{b}x\rangle}{\Longrightarrow} \Longrightarrow\gtrsim \boldsymbol{\nu}\, z\,(\mathrm{Loc}\langle xzy\rangle \mid Q_3) \tag{32}$$

  with

  $$y = \varphi^{-1}(u) \quad \text{ and } \quad \big(Q_3, \varphi \cup (z, ul)\big) \in \mathcal{L}[\![B_4]\!].$$

For $A' \overset{def}{=} B_4$ and $Q' \overset{def}{=} Q_3$ in the assertion of the lemma, this concludes the proof.

The remaining clauses of the lemma can be proved in a similar way. For clauses (1b) and (2b), in order to write the correspondent of (32) one needs the following fact: If $\big(Q,\varphi\big) \in \mathcal{L}[\![A]\!]$ and $\boldsymbol{\nu}\, z\,(\mathrm{Loc}\langle xzy\rangle \mid Q) \Longrightarrow R$, then for some Q' s.t. $Q \Longrightarrow Q'$, we have $R = \boldsymbol{\nu}\, z\,(\mathrm{Loc}\langle xzy\rangle \,|\, Q')$. This follows from Lemmas 7.2 and 7.4 which, together, show that a process $Q \in \mathcal{L}[\![\mathcal{P}_{loc}]\!]$ and its τ-derivatives cannot interact with component $\mathrm{Loc}\langle xzy\rangle$. $\square$

Remark 7.7 Clauses (1b) and (2b) imply that if $\big(P,\varphi\big) \in \mathcal{L}[\![A]\!]$, then P has no weak transitions along locating names. $\square$

8 Cancellation lemmas

This section is devoted to the proof of certain cancellation lemmas which, together with the results operational correspondence for $\mathcal{L}$, will allow us to prove the full abstraction of $\mathcal{S}$ and $\mathcal{L}$ in the next section.

Lemma 8.1 *Suppose $x \notin fn(P,Q)$. Then $\boldsymbol{\nu}\, z\,(\overline{x}\langle z\rangle \,|\, P) \approx \boldsymbol{\nu}\, z\,(\overline{x}\langle z\rangle \,|\, Q)$ implies $P \approx Q$.*

PROOF: We have $\boldsymbol{\nu}\, z\,(\overline{x}\langle z\rangle \,|\, P) \overset{(\boldsymbol{\nu}\, z)\overline{x}\langle z\rangle}{\Longrightarrow} \equiv P$. Since $\boldsymbol{\nu}\, z\,(\overline{x}\langle z\rangle \,|\, P) \approx \boldsymbol{\nu}\, z\,(\overline{x}\langle z\rangle \,|\, Q)$, for some R,

$$\boldsymbol{\nu}\, z\,(\overline{x}\langle z\rangle \,|\, Q) \overset{(\boldsymbol{\nu}\, z)\overline{x}\langle z\rangle}{\Longrightarrow} R \approx P\,.$$

Since $x \notin fn(Q)$, for some Q' s.t. $Q \Longrightarrow Q'$, and Q'', the above transition can be written as

$$\boldsymbol{\nu}\, z\,(\overline{x}\langle z\rangle \,|\, Q) \Longrightarrow \boldsymbol{\nu}\, z\,(\overline{x}\langle z\rangle \,|\, Q') \overset{(\boldsymbol{\nu}\, z)\overline{x}\langle z\rangle}{\longrightarrow} \equiv Q' \Longrightarrow Q''$$

and, by Lemma 4.5, $Q'' \equiv R$. Thus, we have found a Q'' s.t.

$$Q \Longrightarrow Q'' \approx P\,. \tag{33}$$

In a completely symmetric way — just exchange the roles of P and Q, we can find P'' s.t.

$$P \Longrightarrow P'' \approx Q\,. \tag{34}$$

Now, we can derive $P \approx Q$ from (33) and (34), for

$$\{(P,Q)\} \cup \approx$$

is a weak bisimulation. $\square$

The details of the proofs of Lemmas 8.2 and 8.3 below are in Appendix A. They require proving a few properties of parallel compositions of "wires", that is compositions of the form

$$!\, z_1.\, \overline{y_1} \,|\, \ldots \,|\, !\, z_n.\, \overline{y_n}.$$

We recall that $P \in \mathcal{L}[\![\mathcal{P}_{loc}]\!]$ means that there exists a locating function φ and a located process A s.t. $\big(P, \varphi\big) \in \mathcal{L}[\![A]\!]$.

Lemma 8.2 *Let $P, Q \in \mathcal{L}[\![\mathcal{P}_{loc}]\!]$ and z, y be locating names. Then $!\, z.\overline{y} \,|\, P \approx\, !\, z.\overline{y} \,|\, Q$ implies $P \approx Q$.* $\square$

Lemma 8.3 *Let $P_1, P_2 \in \mathcal{L}[\![\mathcal{P}_{loc}]\!]$ and $z \notin \{y, y'\}$. Then*

$$\boldsymbol{\nu}\, z\,(\mathrm{Loc}\langle xzy\rangle \,|\, P_1) \approx \boldsymbol{\nu}\, z\,(\mathrm{Loc}\langle xzy'\rangle \,|\, P_2) \quad \textit{iff} \quad y = y' \textit{ and } P_1 \approx P_2. \qquad \square$$

9 Full abstraction

In the statement of the soundness and completeness of $\mathcal{L}$, remember that by Lemma 5.3 we can assume that the encoding of two located processes use the same locating function.

Theorem 9.1 (soundness of $\mathcal{L}$) *If $\left(P_i, \varphi\right) \in \mathcal{L}[\![A_i]\!]$, $i = 1, 2$ and $P_1 \approx P_2$, then $A_1 \approx_\ell A_2$.*

Proof: We show that

$$\mathcal{R} = \{(A_1, A_2) \text{ s.t. } \left(P_i, \varphi\right) \in \mathcal{L}[\![A_i]\!] \text{ and } P_1 \approx P_2, \text{ for some } P_i \text{ and } \varphi, i = 1, 2\}$$

is a $\approx_\ell$-bisimulation up-to $\equiv$. Let A_i and P_i as in the definition of $\mathcal{R}$.

Case (a) Suppose $A_1 \overset{a\langle\widetilde{b}\rangle}{\underset{ul}{\Longrightarrow}} A_1''$, with $ul \notin \texttt{cod}(\varphi)$. Then, by Lemma 7.6(1a),

$$P_1 \overset{a\langle\widetilde{b}x\rangle}{\Longrightarrow} \gtrsim \boldsymbol{\nu}\, z\,(\mathrm{Loc}\langle xzy\rangle \mid P_1') \overset{def}{=} H_1$$

with $y = \varphi^{-1}(u)$ and $\left(P_1', \varphi \cup (z, ul)\right) \in \mathcal{L}[\![A_1']\!]$, for some $A_1' \equiv A_1''$. Since $P_1 \approx P_2$, for some P_2'',

$$P_2 \overset{a\langle\widetilde{b}x\rangle}{\Longrightarrow} P_2'' \approx H_1 .$$

By Lemma 7.6(1b), we have, for some y', u', P_2' and A_2',

$$P_2'' \gtrsim \boldsymbol{\nu}\, z\,(\mathrm{Loc}\langle xzy'\rangle \mid P_2') \overset{def}{=} H_2 \qquad \text{and} \qquad A_2 \overset{a\langle\widetilde{b}\rangle}{\underset{u'l}{\Longrightarrow}} \equiv A_2'$$

with $y' = \varphi^{-1}(u')$ and $\left(P_2', \varphi \cup (z, u'l)\right) \in \mathcal{L}[\![A_2']\!]$. From $H_1 \approx P_2'' \gtrsim H_2$, we get $H_1 \approx H_2$ which, by Lemma 8.3, implies $y = y'$ and $P_1' \approx P_2'$. Therefore, since φ is injective, also $u = u'$ and, since $\left(P_i', \varphi \cup (z, ul)\right) \in \mathcal{L}[\![A_i']\!]$, $i = 1, 2$, we conclude that $A_1' \,\mathcal{R}\, A_2'$.

Case (b) $A_1 \overset{(\boldsymbol{\nu}\,\widetilde{b'})\overline{a}\langle\widetilde{b}\rangle}{\underset{ul}{\Longrightarrow}} A_1''$.

Similar to case (a).

Case (c) $A_1 \Longrightarrow A_1'$.

This case is easier than case (a) and can be worked out similarly.

$\square$

Theorem 9.2 (completeness of $\mathcal{L}$) *If $\left(P_i, \varphi\right) \in \mathcal{L}[\![A_i]\!]$, $i = 1, 2$, and $A_1 \approx_\ell A_2$, then $P_1 \approx_\ell P_2$.*

PROOF: We show that the relation

$$\mathcal{R} = \{(P_1, P_2) \text{ s.t. } \big(P_i, \varphi\big) \in \mathcal{L}[\![A_i]\!] \text{ and } A_1 \approx_\ell A_2, \text{ for some } A_i \text{ and } \varphi,\ i = 1,2\}$$

is a $\approx$-bisimulation up-to context and up-to $\gtrsim$. Let P_i and A_i, $i = 1,2$, be as in the definition of $\mathcal{R}$. We proceed by a case analysis on the actions which P_1 may perform. We only show the details for the case of weak input transitions, since those of output and silent transitions are similar. Thus suppose $P_1 \stackrel{a\langle\widetilde{bx}\rangle}{\Longrightarrow} P_1''$. By Lemma 7.6(1b), there are A_1', P_1', y, and $ul \notin \mathsf{cod}(\varphi)$ s.t.

$$A_1 \underset{ul}{\stackrel{a\langle\widetilde{b}\rangle}{\Longrightarrow}} \equiv A_1' \qquad \text{and} \qquad P_1'' \gtrsim \boldsymbol{\nu}\, z\, \big(\mathrm{Loc}\langle xzy\rangle \mid P_1'\big),$$

with $y = \varphi^{-1}(u)$ and $\big(P_1', \varphi \cup (z, ul)\big) \in \mathcal{L}[\![A_1']\!]$. Since $A_1 \approx_\ell A_2$, also $A_2 \underset{ul}{\stackrel{a\langle\widetilde{b}\rangle}{\Longrightarrow}} A_2'' \approx_\ell A_1'$. By Lemma 7.6(1a) and injectivity of φ, there are $A_2' \equiv A_2''$ and P_2' s.t.

$$P_2 \stackrel{a\langle\widetilde{bx}\rangle}{\Longrightarrow} \gtrsim \boldsymbol{\nu}\, z\, \big(\mathrm{Loc}\langle xzy\rangle \mid P_2'\big)$$

with $\big(P_2', \varphi \cup (z, ul)\big) \in \mathcal{L}[\![A_2']\!]$. Summarising, there are a static context $C[\cdot] \stackrel{def}{=} \boldsymbol{\nu}\, z\, \big(\mathrm{Loc}\langle xzy\rangle | [\cdot]\big)$ and processes P_i', A_i', $i = 1,2$, s.t.

$$P_i \stackrel{a\langle\widetilde{bx}\rangle}{\Longrightarrow} \gtrsim C[P_i'], \quad A_1' \approx_\ell A_2' \quad \text{and} \quad \big(P_i', \varphi \cup (z, ul)\big) \in \mathcal{L}[\![A_i]\!].$$

The last two conditions gives us $P_1' \mathrel{\mathcal{R}} P_2'$. This is enough, because $\mathcal{R}$ is a $\approx$-bisimulation up-to context and up-to $\gtrsim$. $\square$

The two previous results give the full abstraction of $\mathcal{L}$ and, as a corollary, the full abstraction of $\mathcal{S}$.

Corollary 9.3 (full abstraction of $\mathcal{S}$ w.r.t. $\approx_\ell$) *For all standard processes P_1, P_2 and locating name y, it holds that*

$$P_1 \approx_\ell P_2 \quad \textit{iff} \quad \mathcal{S}[\![P_1]\!]\langle y\rangle \approx \mathcal{S}[\![P_2]\!]\langle y\rangle.$$

PROOF: By definitions of $\mathcal{S}$ and $\mathcal{L}$, we have $\big(\mathcal{S}[\![P_i]\!]\langle y\rangle, \{(y, \emptyset)\}\big) \in \mathcal{L}[\![P_i]\!]$, $i = 1,2$. Then the result follows from Theorems 9.1 and 9.2. $\square$

We use the full abstraction of $\mathcal{S}$ in the next section to show that $\approx_\ell$ is not a congruence, but that the correctness of $\mathcal{S}$ extends to the induced congruence.

10 Applications

Certain algebraic laws for $\approx_\ell$ can be derived as straightforward corollaries from the correctness of the encodings $\mathcal{S}$ and $\mathcal{L}$. Examples are:

- $u :: v :: P \approx_\ell uv :: P$;
- $P \approx_\ell \emptyset :: P$;
- (distributivity of a restricted replication over parallel composition) if a occurs free in P, Q and R only as subject of output prefixes, then $\boldsymbol{\nu}\, a\left(\,!\, a(\tilde{c}).\, R \mid P \mid Q\right) \approx_\ell \boldsymbol{\nu}\, a\left(\,!\, a(\tilde{c}).\, R \mid P\right) \mid \boldsymbol{\nu}\, a\left(\,!\, a(\tilde{c}).\, R \mid Q\right)$.

The last law is inferred from the analogous law for $\approx$, namely Lemma 2.6(9a), since for some R', P' and Q' we have

$$\begin{array}{rcl} \mathcal{S}[\![\boldsymbol{\nu}\, a\left(\,!\, a(\tilde{c}).\, R \mid P \mid Q\right)]\!]\langle y\rangle & = & \boldsymbol{\nu}\, a\left(\,!\, a(\tilde{c}).\, R' \mid P' \mid Q'\right) \\ \mathcal{S}[\![\boldsymbol{\nu}\, a\left(\,!\, a(\tilde{c}).\, R \mid P\right) \mid \boldsymbol{\nu}\, a\left(\,!\, a(\tilde{c}).\, R \mid Q\right)]\!]\langle y\rangle & = & \boldsymbol{\nu}\, a\left(\,!\, a(\tilde{c}).\, R' \mid P'\right) \mid \boldsymbol{\nu}\, a\left(\,!\, a(\tilde{c}).\, R' \mid Q'\right) \end{array}$$

and a occurs free in R', P' and Q' only as subject of output prefixes. Similarly, we can prove for $\approx_\ell$ the analogous of Lemma 2.6(9b), namely the distributivity of a restricted replication over replication. The skeptical reader might want to verify that a direct proof (i.e., without going through the encoding) of these distributivity laws is rather tedious.

Another example of application of $\mathcal{S}$ is provided below.

Example 10.1 *We want to show that*

$$\boldsymbol{\nu}\, b\,(a.\overline{b} \mid b.c) \;+\; \boldsymbol{\nu}\, b\,(c.\overline{b} \mid b.a) \approx_\ell a \mid c\,. \tag{35}$$

Simple equalities like this can be proved by exhibiting the appropriate $\approx_\ell$-bisimulation. But it is rather hard to device a general set of laws for $\approx_\ell$ from which to infer the equalities with purely equational reasoning. Indeed, Boudol et al. *[BCHK91b], to give an axiomatisation of $\approx_\ell$ on finite CCS terms, enrich the language with a new operator which allows, among other things, to define an expansion-like law for $\approx_\ell$.*

Here we show how we can prove (35) using our encoding and the algebraic theory of the interleaving bisimilarity $\approx$. First, we consider the process $\mathcal{S}[\![\boldsymbol{\nu}\, b\,(a.\overline{b} \mid b.c)]\!]\langle y\rangle$. We have

$$\mathcal{S}[\![a.\overline{b}]\!]\langle y\rangle = a(x).\, P\,, \quad \text{for } P \stackrel{def}{=} \boldsymbol{\nu}\, z\,(\mathrm{Loc}\langle xzy\rangle \mid \mathcal{S}[\![\overline{b}]\!]\langle z\rangle)\;,$$

and $\mathcal{S}[\![b.c]\!]\langle z\rangle = b(x).\,Q$*, for some* Q*, and we can derive:*

$$
\begin{array}{rcl}
\mathcal{S}[\![\boldsymbol{\nu}\, b\,(a.\,\overline{b} \mid b.c)]\!]\langle y\rangle & = & \\
\boldsymbol{\nu}\, b\,(a(x).\,P \mid b(x).\,Q) & \approx & (36)\\
\boldsymbol{\nu}\, b\Big(a(x).\,(P \mid b(x).\,Q) \;+\; b(x).\,(a(x).\,P \mid Q)\Big) & \approx & (37)\\
a(x).\,\boldsymbol{\nu}\, b\,(P \mid b(x).\,Q) & = & (38)\\
a(x).\,\boldsymbol{\nu}\, b\Big(\boldsymbol{\nu}\, z\,(\mathrm{Loc}\langle xzy\rangle \mid \mathcal{S}[\![\overline{b}]\!]\langle z\rangle) \mid \mathcal{S}[\![b.c]\!]\langle y\rangle\Big) & \equiv & (39)\\
a(x).\,\boldsymbol{\nu}\, z\Big(\mathrm{Loc}\langle xzy\rangle \mid \boldsymbol{\nu}\, b\,(\mathcal{S}[\![\overline{b}]\!]\langle z\rangle \mid \mathcal{S}[\![b.c]\!]\langle y\rangle)\Big) & \approx & (40)\\
a(x).\,\boldsymbol{\nu}\, z\,(\mathrm{Loc}\langle xzy\rangle \mid \mathcal{S}[\![c]\!]\langle y\rangle)\,. & &
\end{array}
$$

Equality (36) is obtained from the expansion law (Lemma 2.6(4)); (37) from laws (1-3) of Lemma 2.6, which relate restriction to sum and prefixing. The step (38) is obtained by unfolding the definition of P *and folding the one of* $b(x).Q$*. Equality (39) uses* $\equiv$ *to change the scope of the restrictions at* z *and* b*. Finally, (40) is derived from the equality*

$$\boldsymbol{\nu}\, b\,(\mathcal{S}[\![\overline{b}]\!]\langle z\rangle \mid \mathcal{S}[\![b.c]\!]\langle y\rangle) \approx \mathcal{S}[\![c]\!]\langle y\rangle \tag{41}$$

for whose proof we refer to (9), in Example 5.4. Thus, (36)-(40) yield

$$\mathcal{S}[\![\boldsymbol{\nu}\, b\,(a.\overline{b} \mid b.c)]\!]\langle y\rangle \approx\; a(x).\,\boldsymbol{\nu}\, z\,(\mathrm{Loc}\langle xzy\rangle \mid \mathcal{S}[\![c]\!]\langle y\rangle\,. \tag{42}$$

Similarly — just exchange the roles of a *and* c *— we can prove*

$$\mathcal{S}[\![\boldsymbol{\nu}\, b\,(c.\overline{b} \mid b.a)]\!]\langle y\rangle \approx\; c(x).\,\boldsymbol{\nu}\, z\,(\mathrm{Loc}\langle xzy\rangle \mid \mathcal{S}[\![a]\!]\langle y\rangle)\,. \tag{43}$$

Since all processes in (42) and (43) are stable, by Proposition 2.3 we can sum the two equalities to get

$$
\begin{array}{c}
\mathcal{S}[\![\boldsymbol{\nu}\, b\,(a.\overline{b} \mid b.c)]\!]\langle y\rangle \;+\; \mathcal{S}[\![\boldsymbol{\nu}\, b\,(c.\overline{b} \mid b.a)]\!]\langle y\rangle \quad \approx \qquad (44)\\
a(x).\,\boldsymbol{\nu}\, z\,(\mathrm{Loc}\langle xzy\rangle \mid \mathcal{S}[\![c]\!]\langle y\rangle \;+\; c(x).\,\boldsymbol{\nu}\, z\,(\mathrm{Loc}\langle xzy\rangle \mid \mathcal{S}[\![a]\!]\langle y\rangle)\,.
\end{array}
$$

Now, we can conclude. We have

$$
\begin{array}{rcl}
\mathcal{S}[\![a \mid c]\!]\langle y\rangle & = & \\
\mathcal{S}[\![a]\!]\langle y\rangle \mid \mathcal{S}[\![c]\!]\langle y\rangle & \approx & \\
& & \textit{(expansion law and rules for } \equiv\textit{)}\\
a(x).\,\boldsymbol{\nu}\, z\,(\mathrm{Loc}\langle xzy\rangle \mid \mathcal{S}[\![c]\!]\langle y\rangle) \;+\; c(x).\,\boldsymbol{\nu}\, z\,(\mathrm{Loc}\langle xzy\rangle \mid \mathcal{S}[\![a]\!]\langle y\rangle) & \approx & \textit{(from (44))}\\
\mathcal{S}[\![\boldsymbol{\nu}\, b\,(a.\overline{b} \mid b.c)]\!]\langle y\rangle \;+\; \mathcal{S}[\![\boldsymbol{\nu}\, b\,(c.\overline{b} \mid b.a)]\!]\langle y\rangle & = & \\
\mathcal{S}[\![\boldsymbol{\nu}\, b\,(a.\overline{b} \mid b.c) \;+\; \boldsymbol{\nu}\, b\,(c.\overline{b} \mid b.a)]\!]\langle y\rangle & &
\end{array}
$$

from which (35) follows by the full abstraction of $\mathcal{S}$*.*

We now use $\mathcal{S}$ and $\mathcal{L}$ to prove the congruence properties of $\approx_\ell$.

Theorem 10.2 *The relation $\approx_\ell$ is preserved by the operators of parallel composition, output prefix, stable sum, matching, restriction and location prefix.*

PROOF: Use the definitions of the encodings $\mathcal{S}$ and $\mathcal{L}$ and their full abstraction, and the congruence properties of $\approx$ in Proposition 2.3. $\square$

The above theorem does not mention non-stable sum and input prefix, for which $\approx$ is not a congruence. It is easy to see that the problems of $\approx$ with sum are reflected back onto $\approx_\ell$: For instance, $\tau.b \approx_\ell b$, but $a + \tau.b \not\approx_\ell a + b$.

The situation for input prefix is more intriguing. Asking whether a relation is preserved by input prefix amounts to asking whether the relation is preserved by name substitution. Now, it is easy to see that things can go wrong because of the matching operator: If $a \neq b$, then $[a = b]a.\mathbf{0}$ is equivalent to $\mathbf{0}$, but $([a = b]a.\mathbf{0})\{a/b\}$ has an a-transition and hence is *not* equivalent to $\mathbf{0}\{a/b\}$. But at least, is $\approx_\ell$ substitutive on processes which do not contain matching? This is a reasonable question, on the one hand because in practice the matching operator can often be avoided, on the other hand because the typical examples which show the non-substitutivity of an interleaving equivalence in absence of matching involve the expansion law, not valid in non-interleaving semantics. A positive answer to this question would have two immediate and influential consequences. Firstly, the verification of process bisimilarities would not have the state explosion problem caused by input prefix (as it happens with $\approx$): To check whether $a(x).P$ and $a(x).Q$ are equivalent, we would not have to consider *all* possible instantiations of the bound name x, but a single instantiation, with a fresh name, would be enough. Secondly, we would not have the proliferation of different forms of bisimulation — like the late, early and open of the interleaving semantics [PS93, San93] — since such a variety is precisely caused by the sensitivity to name instantiation in the input clause. Unfortunately, the answer is negative. To see this, take processes

$$P \stackrel{def}{=} \boldsymbol{\nu}\, b\,(a.\overline{b} \mid b.\overline{c}) + \boldsymbol{\nu}\, b\,(\overline{c}.\overline{b} \mid b.a) \qquad \text{and} \qquad Q \stackrel{def}{=} a \mid \overline{c}.$$

Reasoning as in Example 10.1, we can prove $P \approx_\ell Q$; however we have $P\{a/c\} \not\approx_\ell Q\{a/c\}$, since $Q\{a/c\} \longrightarrow\equiv \mathbf{0}$, which $P\{a/c\}$ cannot match. (Intuitively, the failure arises because location bisimulation does not coincide with its multistep version). Note that, however, P and Q are not *causal bisimilar* [DD89]: In the summands of P there is a temporal dependency between the actions at a and c, whereas these actions are unrelated in Q. Hence the above example still leaves open the corresponding congruence question for causal bisimulation.

Having shown that $\approx_\ell$ is not preserved by sum and input prefix, we want to consider its induced congruence, as we did with $\approx$.

Definition 10.3 (location congruence) *Two located processes A and B are* location congruent*, written $A \approx_\ell^c B$, if for all substitution σ*

1. *whenever $A\sigma \stackrel{\tau}{\Longrightarrow} A'$, there exists B' s.t. $B\sigma \stackrel{\tau}{\Longrightarrow} B'$ and $A' \approx_\ell B'$;*
2. *whenever $A\sigma \underset{ul}{\stackrel{\mu}{\Longrightarrow}} A'$ and $l \notin \mathcal{A}t(A, B)$, there exists B' s.t. $B\sigma \underset{ul}{\stackrel{\mu}{\Longrightarrow}} B'$ and $A' \approx_\ell B'$.*

and the converse of (1) and (2), on the actions which $B\sigma$ can perform. $\square$

The correctness of the encoding $\mathcal{S}$ extends to $\approx_\ell^c$:

Theorem 10.4 (full abstraction of $\mathcal{S}$ for $\approx_\ell^c$ and $\approx^c$) *For all standard processes P_1, P_2 and locating name y, it holds that*

$$P_1 \approx_\ell^c P_2 \quad \textit{iff} \quad \mathcal{S}[\![P_1]\!]\langle y\rangle \approx^c \mathcal{S}[\![P_2]\!]\langle y\rangle.$$

PROOF: We consider the implication form right to left, i.e. $\mathcal{S}[\![P_1]\!]\langle y\rangle \approx^c \mathcal{S}[\![P_2]\!]\langle y\rangle$ implies $P_1 \approx_\ell^c P_2$; the opposite one can be handled similarly.

By Lemma 7.1(1), on standard processes, $\mathcal{S}$ and substitution commute, i.e. $\mathcal{S}[\![P]\!]\langle y\rangle\sigma = \mathcal{S}[\![P\sigma]\!]\langle y\rangle$. As a consequence of this and Corollary 9.3, it is enough to prove the clauses of Definition 10.3 when σ is the identity. Now, clause (2) follows from the fact that $P_1 \approx_\ell P_2$ as can be derived from $\mathcal{S}[\![P_1]\!]\langle y\rangle \approx \mathcal{S}[\![P_2]\!]\langle y\rangle$ and Corollary 9.3. For clause (1), use Lemmas 7.4 and 7.5 and the hypothesis $\mathcal{S}[\![P_1]\!]y \approx^c \mathcal{S}[\![P_2]\!]y$. $\square$

We can use the full abstraction of $\mathcal{S}$ w.r.t. $\approx_\ell^c$ to investigate its congruence properties, as we did in Theorem 10.2 with $\approx_\ell$.

Theorem 10.5 *The relation $\approx_\ell^c$ is preserved by all operators of the language.* $\square$

A standard argument shows that $\approx_\ell^c$ is indeed the largest congruence contained in $\approx_\ell$.

11 Conclusions

In this paper we have begun the study of non-interleaving semantics in calculi for mobile processes like π-calculus. We have not investigated new ideas of behavioural equivalence. This might be a fruitful direction of research, given the novel features of π-calculus w.r.t.

"traditional" process algebras — first of all, the primary role of name instantiation in the operational semantics. Our interest was rather for well-known behavioural equivalences. We believe that a few of them have natural and elegant formalisations in π-calculus.

We have concentrated on the spatial-sensitive equivalences, in particular on location bisimulation ($\approx_\ell$) [BCHK91b], in our view one of the most convincing in this group. Our main result is that $\approx_\ell$ can be expressed within the ordinary interleaving observation equivalence ($\approx$). The possibility of encoding $\approx_\ell$ does not mean that it is superfluous. It represents an important behavioural equivalence, and with a definite practical relevance. The descriptions in terms of $\approx_\ell$ and $\approx$ complement each other: The former is more abstract and intuitive; the latter is more concrete and — often — easier to reason about.

The major technical problem in the definition of the encoding was the modeling of silent transitions, which in location bisimulation are computed with a transition system different from the one for visible actions. Indeed, the treatment of τ-actions is precisely where spatiality- and causality-based equivalences differ [Kie91]. We have experimented various variations of the encoding; we think that it would be difficult to improve its efficiency without losing the full abstraction.

Our encoding does not preserve process finiteness. That is, processes which do not contain constants in their syntactic form and hence can only perform a finite number of actions, are in general mapped onto processes which do not enjoy the same property. In consequence, an axiomatisation of observation congruence on the finite processes does not yield a sound and complete proof system for location congruence. The reason finiteness is not preserved is the introduction of the components $!\,z.\,\overline{y}$, in the encoding of prefixes. These components are used as wires, which transform outputs at z into outputs at y, and are fundamental for the correct modeling of silent actions. In particular, we can derive the equation

$$\boldsymbol{\nu}\, z\,(\,!\,z.\,\overline{y} \mid P) \gtrsim P\{y/z\} \tag{45}$$

of Proposition 6.2. Clearly, a wire should be allowed an infinite behaviour, for in general we cannot determine a bound on the number of messages which have to traverse it. However, such a bound should exist in the case of finite processes. Hence it is reasonable to hope that there are modifications of the encoding which, on finite processes, are fully abstract and yet return finite processes. The issue appears non-trivial though, for properties of wires like (45) are used in various critical points of our proofs. An alternative is to leave the encoding as it is and give a proof system for location congruence from appropriate laws for observation congruence plus the lemmas of Sections 6 and 8 (say, Corollaries 6.3 and 6.4 — in terms of $\approx$ — and the cancellation lemmas) to manipulate the components implementing the location machinery (something like the way we handled Example 10.1).

Recently, we have undertaken a study similar to the one in this paper for causal equivalences. We hope to derive an encoding for causal bisimulation [DD89] with very few modifications of the one for location bisimulation used in this paper; some intuitions about how this might be achieved have been given in Section 5. The encodings would provide us with a means of relating and contrasting spatial- and causal-sensitive equivalences.

Various proofs in this paper have been made possible or facilitated by the use of the expansion relation or/and "bisimulation up-to" techniques. We think that the development of these or similar mathematical tools is an important direction to pursue in calculi for mobile processes and in higher-order calculi.

References

[Ace91] L. Aceto. A static view of localities. Rapport de recherche 1483, INRIA-Sophia Antipolis, 1991. To appear in *Formal Aspects of Computing*.

[AKH92] S. Arun-Kumar and M. Hennessy. An efficiency preorder for processes. *Acta Informatica*, 29:737–760, 1992.

[BCHK91a] G. Boudol, I. Castellani, M. Hennessy, and A. Kiehn. Observing localities. TR 4/91, University of Sussex, 1991. To appear in *Theoretical Computer Science*. Extended Abstract in Proc. MFCS 91, LNCS 520, 1991.

[BCHK91b] G. Boudol, I. Castellani, M. Hennessy, and A. Kiehn. A theory of processes with localities. TR 13/91, University of Sussex, 1991. To appear in *Formal Aspects of Computing*. Extended Abstract in Proc. CONCUR 92, 1992.

[Cas93] I. Castellani. Observing distribution in processes. In *Proc. MFCS 93*, volume 711, pages 321–331. Springer Verlag, 1993.

[CD93] F. Corradini and R. De Nicola. Locality and causality in distributed process algebras. Report SI/R/R-93/05, Dipartimento di Scienze dell'Informazione, Università degli studi di Roma "La Sapienza", 1993.

[DD89] P. Degano and P. Darondeau. Causal trees. In *15th ICALP*, volume 372 of *Lecture Notes in Computer Science*, pages 234–248. Springer Verlag, 1989.

[DDM87] P. Degano, R. De Nicola, and U. Montanari. Observational equivalences for concurrency models. In *Proc. 3rd IFIP WG 2.2 working conference*, pages 105–129. North Holland, 1987.

[DDM93] P. Degano, R. De Nicola, and U. Montanari. Universal axioms for bisimulation. *Theoretical Computer Science*, 114:63–91, 1993.

[DP92] P. Degano and C. Priami. Proved trees. In W. Kuich, editor, *Proc. ICALP 92*, volume 623 of *Lecture Notes in Computer Science*. Springer Verlag, 1992.

[Eng93] J. Engelfriet. A multiset semantics for the π-calculus with replication. In E. Best, editor, *Proceedings of CONCUR '93*, volume 715, pages 7–21. Springer Verlag, 1993.

[GL91] R. Gorrieri and C. Laneve. The limit of split_n-bisimulations for CCS agents. In *Proc. MFCS 91*, volume 520 of *Lecture Notes in Computer Science*, pages 170–180. Springer Verlag, 1991.

[GV87] R.J. van Glabbeek and F.W. Vaandrager. Petri nets models for algebraic theories of concurrency. In *Proc. PARLE Conference*, volume 259 of *Lecture Notes in Computer Science*, pages 224–242. Springer Verlag, 1987.

[Kie91] A. Kiehn. Local and global causes. Technical Report Report 342/23/91, Technische Universität München, 1991.

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

[Mil91] R. Milner. The polyadic π-calculus: a tutorial. Technical Report ECS–LFCS–91–180, LFCS, Dept. of Comp. Sci., Edinburgh Univ., October 1991. Also in *Logic and Algebra of Specification*, ed. F.L. Bauer, W. Brauer and H. Schwichtenberg, Springer Verlag, 1993.

[Mil92] R. Milner. Functions as processes. *Journal of Mathematical Structures in Computer Science*, 2(2):119–141, 1992.

[MN92] M. Mukund and M. Nielsen. CCS, locations and asynchronous transition systems. In *Proc. FST-TCS 92*, volume 652 of *Lecture Notes in Computer Science*, pages 328–341. Springer Verlag, 1992.

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

[Mur93] D. Murphy. Observing located concurrency. In *Proc. MFCS 93*, volume 711 of *Lecture Notes in Computer Science*, pages 566–576. Springer Verlag, 1993.

[MY92] M. Montanari and D. Yankelevich. A parametric approach to localities. In W. Kuich, editor, *Proc. ICALP 92*, volume 623 of *Lecture Notes in Computer Science*, pages 617–628. Springer Verlag, 1992.

[PS93] J. Parrow and D. Sangiorgi. Algebraic theories for name-passing calculi. Technical Report ECS–LFCS–93–262, LFCS, Dept. of Comp. Sci., Edinburgh Univ., 1993. Short version to appear in the Proc. REX Summer School 1993, LNCS, Springer Verlag.

[San92] D. Sangiorgi. *Expressing Mobility in Process Algebras: First-Order and Higher-Order Paradigms.* PhD thesis CST–99–93, Department of Computer Science, University of Edinburgh, 1992. Also published as ECS–LFCS–93–266.

[San93] D. Sangiorgi. A theory of bisimulation for the π-calculus. Technical Report ECS–LFCS–93–270, LFCS, Dept. of Comp. Sci., Edinburgh Univ., 1993. Extended Abstract in *Proc. CONCUR '93*, volume 715 of Lecture Notes in Computer Science, Springer Verlag.

[San94] D. Sangiorgi. Bisimulations up to context (provisory title). In preparation, 1994. LFCS, Edinburgh.

[SM92] D. Sangiorgi and R. Milner. The problem of "Weak Bisimulation up to". In W.R. Cleveland, editor, *Proceedings of CONCUR '92*, volume 630, pages 32–46. Springer Verlag, 1992.

[Wal93] D. Walker. Objects in the π-calculus. *Information and Computation*, 1993. To appear.

A Proofs of Lemmas 8.2 and 8.3

We write $\prod_{i=1}^{n} P_i$ to abbreviate $P_1 \mid \cdots \mid P_n$. We give "$\prod$" precedence over "$|$"; therefore $\prod_{i=1}^{n} P_i \mid Q$ means $(\prod_{i=1}^{n} P_i) \mid Q$. We also abbreviate a product of wires $\prod_{i=1}^{n} !\, z_i.\, \overline{y_i}$ as $\prod_{i=1}^{n} W_i$.

We say that $\prod_{i=1}^{n} W_i$ is *unidirectional* if it possible to establish an order "$<$" on the free names of $\prod_{i=1}^{n} W_i$ s.t. for each $W_i = !\, z_i.\, \overline{y_i}$, it holds that $z_i < y_i$. Intuitively, if $\prod_{i=1}^{n} W_i$ is unidirectional, then any interrogation at some name z can only propagate through names of progressively increasing order; in particular, the answer cannot occur at z.

Remark A.1 Suppose $\prod_{i=1}^{n} W_i$ is unidirectional and $z \notin fn(\prod_{i=1}^{n} W_i)$. Then, for any y, it holds that $\prod_{i=1}^{n} W_i \mid !\, z.\, \overline{y}$ is unidirectional (name z can be taken as the bottom element in the order of the free names of $\prod_{i=1}^{n} W_i \mid !\, z.\, \overline{y}$). $\square$

We write $P \stackrel{\mu}{\Longrightarrow}$ if there is P' s.t. $P \stackrel{\mu}{\Longrightarrow} P'$.

Lemma A.2 *Suppose $\prod_{i=1}^{n} W_i$ is unidirectional and $y \neq y'$. Then at least one of the two following transitions is impossible:*

- $\overline{y} \mid \prod_{i=1}^{n} W_i \stackrel{y'}{\Longrightarrow}$;
- $\overline{y'} \mid \prod_{i=1}^{n} W_i \stackrel{y}{\Longrightarrow}$.

PROOF: The assertion is trivial if one of names y or y' is not free in $\prod_{i=1}^{n} W_i$. Suppose both names are free in $\prod_{i=1}^{n} W_i$, and according to the order of the free names of $\prod_{i=1}^{n} W_i$ (as by definition of unidirectionality), it holds that $y < y'$. Then $\overline{y'} \mid \prod_{i=1}^{n} W_i \stackrel{y}{\Longrightarrow}$ cannot hold, for an interrogation at y' can only propagate in $\prod_{i=1}^{n} W_i$ through names whose order is greater than y's order. $\square$

Lemma A.3 *Suppose P and $Q \in \mathcal{L}[\![\mathcal{P}_{loc}]\!]$, $\prod_{i=1}^{n} W_i$ is unidirectional, z, y and y' are locating names and $z \notin fn(\prod_{i=1}^{n} W_i) \cup \{y, y'\}$. Then*

$$\prod_{i=1}^{n} W_i \mid !\, z.\, \overline{y} \mid P \approx \prod_{i=1}^{n} W_i \mid !\, z.\, \overline{y'} \mid Q \quad \textit{implies} \quad y \approx y' \ .$$

PROOF: We reason by contradiction. Suppose $y \neq y'$. By Lemma A.2, $\prod_{i=1}^{n} W_i \mid \overline{y} \stackrel{y'}{\Longrightarrow}$ or $\prod_{i=1}^{n} W_i \mid \overline{y'} \stackrel{y}{\Longrightarrow}$ is impossible. Suppose the latter is impossible. We have

$$\prod_{i=1}^{n} W_i \mid !\, z.\, \overline{y} \mid P \stackrel{z}{\Longrightarrow} \stackrel{\overline{y}}{\Longrightarrow} \ . \tag{46}$$

By contrast, process $\prod_{i=1}^{n} W_i \mid !\, z.\, \overline{y'} \mid Q$, after receiving at z, cannot answer at y. Process $\prod_{i=1}^{n} W_i \mid \overline{y'} \mid !\, z.\, \overline{y'} \mid Q$ does not have a transition $\stackrel{\overline{y}}{\Longrightarrow}$, for the following reasons. Since $Q \in \mathcal{L}[\![\mathcal{P}_{loc}]\!]$, by Remark 7.7 it cannot interact with the component $\prod_{i=1}^{n} W_i \mid \overline{y'} \mid !\, z.\, \overline{y'}$. By the previous assumption, process $\prod_{i=1}^{n} W_i \mid \overline{y'}$ does not have the transition $\stackrel{\overline{y}}{\Longrightarrow}$. Since $z \notin fn(\prod_{i=1}^{n} W_i \mid \overline{y}\,)$, neither $\prod_{i=1}^{n} W_i \mid \overline{y} \mid !\, z.\, \overline{y'}$ has such a transition.

In consequence, $\prod_{i=1}^{n} W_i \mid !\, z.\, \overline{y'} \mid Q$ cannot match the actions in (46), in contradiction with the hypothesis that $\prod_{i=1}^{n} W_i \mid !\, z.\, \overline{y} \mid P \approx \prod_{i=1}^{n} W_i \mid !\, z.\, \overline{y'} \mid Q$. We conclude that it must be $y = y'$. $\square$

Lemma A.4 *Suppose $Q \in \mathcal{L}[\![\mathcal{P}_{loc}]\!]$ and $\prod_{i=1}^{n} W_i \mid Q \stackrel{\mu}{\Longrightarrow} R$, where the subject of μ is not a locating name. Then there are $Q' \in \mathcal{L}[\![\mathcal{P}_{loc}]\!]$, y', and $x \in n(\mu)$ s.t. $Q \stackrel{\mu}{\Longrightarrow} \gtrsim \boldsymbol{\nu}\, z\, (\mathrm{Loc}\langle xzy' \rangle \mid Q')$ and $R \gtrsim \prod_{i=1}^{n} W_i \mid \boldsymbol{\nu}\, z\, (\mathrm{Loc}\langle xzy' \rangle \mid Q')$.*

Proof: The weak transition $\prod_{i=1}^{n} W_i \mid Q \stackrel{\mu}{\Longrightarrow} R$ can be decomposed as

$$\prod_{i=1}^{n} W_i \mid Q \Longrightarrow R_1 \stackrel{\mu}{\longrightarrow} R_2 \Longrightarrow R \tag{47}$$

for some R_1 and R_2. We show that these actions are caused by the subterm Q alone. For this, we analyse each of the three steps in (47). First of all, by Remark 7.7, we know that Q cannot perform actions at a locating name z_i, hence no interaction between Q and $\prod_{i=1}^{n} W_i$ is possible. It follows that $R_1 \stackrel{def}{=} \prod_{i=1}^{n} W_i \mid Q_1$, for some Q_1 s.t. $Q \Longrightarrow Q_1$. Moreover, by Lemma 7.5(2), we have

$$Q_1 \gtrsim Q_2 \tag{48}$$

for some $Q_2 \in \mathcal{L}[\![\mathcal{P}_{loc}]\!]$, and therefore, by the congruence properties of $\gtrsim$, also

$$\prod_{i=1}^{n} W_i \mid Q_1 \gtrsim \prod_{i=1}^{n} W_i \mid Q_2 \,.$$

Now the second step in (47), namely $R_1 = \prod_{i=1}^{n} W_i \mid Q_1 \stackrel{\mu}{\longrightarrow} R_2$. Since the subject of μ is not a locating name z_i,

$$R_2 = \prod_{i=1}^{n} W_i \mid Q_3$$

for some Q_3 s.t. $Q_1 \stackrel{\mu}{\longrightarrow} Q_3$. From (48) we get, for some Q_4,

$$Q_2 \stackrel{\mu}{\longrightarrow} Q_4 \lesssim Q_3 \tag{49}$$

and hence

$$\prod_{i=1}^{n} W_i \mid Q_2 \stackrel{\mu}{\longrightarrow} \prod_{i=1}^{n} W_i \mid Q_4 \lesssim \prod_{i=1}^{n} W_i \mid Q_3 \,. \tag{50}$$

Further, by Lemma 7.2, since $Q_2 \in \mathcal{L}[\![\mathcal{P}_{loc}]\!]$, for some $Q_5 \in \mathcal{L}[\![\mathcal{P}_{loc}]\!]$, y', and $x \in n(\mu)$

$$Q_4 \equiv \boldsymbol{\nu}\, z\, (\mathrm{Loc}\langle xzy'\rangle \mid Q_5)\,. \tag{51}$$

Finally, the third step in (47), namely

$$R_2 = \prod_{i=1}^{n} W_i \mid Q_3 \Longrightarrow R\,. \tag{52}$$

By (50), we have

$$\prod_{i=1}^{n} W_i \mid Q_4 \Longrightarrow R_3 \lesssim R\,. \tag{53}$$

Names z and x in (51) are not free in $\prod_{i=1}^{n} W_i$ and, by Remark 7.7, Q_5 cannot perform actions at a locating name z_i, nor — for analogous reasons — can it interact with $\mathrm{Loc}\langle xzy'\rangle$. We deduce that the action in (53) can be rewritten as

$$\prod_{i=1}^{n} W_i \mid Q_4 \equiv \prod_{i=1}^{n} W_i \mid \boldsymbol{\nu}\, z\, (\mathrm{Loc}\langle xzy'\rangle \mid Q_5) \Longrightarrow \prod_{i=1}^{n} W_i \mid \boldsymbol{\nu}\, z\, (\mathrm{Loc}\langle xzy'\rangle \mid Q_6) \equiv R_3 \tag{54}$$

for some Q_6 s.t.

$$Q_5 \Longrightarrow Q_6\,. \tag{55}$$

Note that it also holds that

$$Q_4 \Longrightarrow\equiv \boldsymbol{\nu}\, z\, (\mathrm{Loc}\langle xzy'\rangle \mid Q_6)\,. \tag{56}$$

Moreover, from (55), since $Q_5 \in \mathcal{L}[\![\mathcal{P}_{loc}]\!]$, by Lemma 7.5 there is $Q_7 \in \mathcal{L}[\![\mathcal{P}_{loc}]\!]$ s.t.

$$Q_6 \gtrsim Q_7 \tag{57}$$

and hence from (53), (54) and (57) we get

$$R \gtrsim R_3 \gtrsim \prod_{i=1}^{n} W_i \ \mid \boldsymbol{\nu}\, z\, (\mathrm{Loc}\langle xzy'\rangle \mid Q_7)\,. \tag{58}$$

Also, from (49), (56) and (57), for some Q_8,

$$Q_3 \Longrightarrow Q_8 \gtrsim \boldsymbol{\nu}\, z\, (\mathrm{Loc}\langle xzy'\rangle \mid Q_7)\,.$$

Let us summarise what we have obtained from the analysis of the three steps in (47). We have found that

$$Q \Longrightarrow Q_1 \xrightarrow{\mu} Q_3 \Longrightarrow Q_8 \gtrsim \boldsymbol{\nu}\, z\, (\mathrm{Loc}\langle xzy'\rangle \mid Q_7) \tag{59}$$

with $Q_7 \in \mathcal{L}[\![\mathcal{P}_{loc}]\!]$, and that

$$\prod_{i=1}^{n} W_i \,|\, Q \stackrel{\mu}{\Longrightarrow} R \;\gtrsim\; \prod_{i=1}^{n} W_i \,|\, \boldsymbol{\nu}\, z\, (\mathrm{Loc}\langle xzy'\rangle \,|\, Q_7)\,. \tag{60}$$

For $Q' \stackrel{def}{=} Q_7$ in the assertion of the lemma, this concludes the proof. $\square$

Now we are ready to prove Lemma 8.2. We first recall its assertion:

Let $P, Q \in \mathcal{L}[\![\mathcal{P}_{loc}]\!]$ and z, y be locating names. Then $!\, z.\, \overline{y} \,|\, P \,\approx\, !\, z.\, \overline{y} \,|\, Q$ implies $P \,\approx\, Q$.

Proof of Lemma 8.2.

We prove that

$$\begin{array}{lll} \mathcal{R} \;=\{\,(P,Q) & \text{s.t.} & P,Q \in \mathcal{L}[\![\mathcal{P}_{loc}]\!] \text{ and for some } n, \text{ and locating names } z_i, y_i \\ & & \text{with } \prod_{i=1}^{n} !\, z_i.\, \overline{y_i} \text{ unidirectional, it holds that} \\ & & \prod_{i=1}^{n} !\, z_i.\, \overline{y_i} \,|\, P \,\approx\, \prod_{i=1}^{n} !\, z_i.\, \overline{y_i} \,|\, Q\,\} \end{array}$$

is a $\approx$-bisimulation up-to context and up-to $\gtrsim$. Take $(P,Q) \in \mathcal{R}$. Then $P, Q \in \mathcal{L}[\![\mathcal{P}_{loc}]\!]$ and $\prod_{i=1}^{n} !\, z_i.\, \overline{y_i} | P \,\approx\, \prod_{i=1}^{n} !\, z_i.\, \overline{y_i} | Q$. Suppose $P \stackrel{\mu}{\Longrightarrow} P''$. We have to find $P', Q' \in \mathcal{L}[\![\mathcal{P}_{loc}]\!]$, a static context $C[\cdot]$, and a process H of the form $\prod_{j=1}^{m} !\, z_j.\, \overline{y_j}$ and unidirectional s.t.

$$P'' \gtrsim C[P'], \quad Q \stackrel{\widehat{\mu}}{\Longrightarrow} \gtrsim C[Q'] \quad \text{and} \quad H \,|\, P' \,\approx\, H \,|\, Q'. \tag{61}$$

In the following, we suppose $\mu \neq \tau$; the case $\mu = \tau$ is simpler. We abbreviate $\prod_{i=1}^{n} !\, z_i.\, \overline{y_i}$ as $\prod_{i=1}^{n} W_i$. Since $P \in \mathcal{L}[\![\mathcal{P}_{loc}]\!]$, we can use Lemma 7.6 and from $P \stackrel{\mu}{\Longrightarrow} P''$ we get

$$P'' \gtrsim \boldsymbol{\nu}\, z\, (\mathrm{Loc}\langle xzy\rangle \,|\, P') \equiv \boldsymbol{\nu}\, z\, (\overline{x}\langle z\rangle \,|\; !\, z.\, \overline{y} \,|\, P') \tag{62}$$

for some y and $P' \in \mathcal{L}[\![\mathcal{P}_{loc}]\!]$. Moreover, using the rule `S-PAR`, we can infer

$$\begin{array}{rcl} \prod_{i=1}^{n} W_i \,|\, P & \stackrel{\mu}{\Longrightarrow} & \prod_{i=1}^{n} W_i \,|\, P'' \\ & \gtrsim & \prod_{i=1}^{n} W_i \,|\, \boldsymbol{\nu}\, z\, (\overline{x}\langle z\rangle \,|\; !\, z.\, \overline{y} \,|\, P') \\ & \equiv & \boldsymbol{\nu}\, z\, (\overline{x}\langle z\rangle \,|\; !\, z.\, \overline{y} \,|\, \prod_{i=1}^{n} W_i \,|\, P')\,. \end{array} \tag{63}$$

Thus, we have found the P' to use in (61); we also set

$$C[\cdot] \stackrel{def}{=} \boldsymbol{\nu}\, z\, (\overline{x}\langle z\rangle \,|\; !\, z.\, \overline{y} \,|\, [\cdot]) \qquad \text{and} \qquad H \stackrel{def}{=} \,!\, z.\, \overline{y} \,|\, \prod_{i=1}^{n} W_i\,.$$

Since $z \notin fn(\prod_{i=1}^n W_i)$, Remarks A.1 tells us that H is unidirectional. It remains to find Q' and prove the remaining requirements in (61). Given (63), and since $\prod_{i=1}^n W_i \mid P \approx \prod_{i=1}^n W_i \mid Q$, there is R s.t.

$$\prod_{i=1}^n W_i \mid Q \stackrel{\mu}{\Longrightarrow} R \approx \boldsymbol{\nu}\, z\, (\overline{x}\langle z\rangle \mid !\, z.\, \overline{y} \mid \prod_{i=1}^n W_i \mid P')\,. \tag{64}$$

The subject of μ is not a locating name. We can apply Lemma A.4 and infer that there are $Q' \in \mathcal{L}[\![\mathcal{P}_{loc}]\!]$ and y' s.t.

$$Q \stackrel{\mu}{\Longrightarrow} \gtrsim \boldsymbol{\nu}\, z\, (\mathrm{Loc}\langle xzy'\rangle \mid Q') \tag{65}$$

and

$$\begin{aligned} R \;&\gtrsim\; \prod_{i=1}^n W_i \mid \boldsymbol{\nu}\, z\, (\mathrm{Loc}\langle xzy'\rangle \mid Q') \\ &\equiv\; \boldsymbol{\nu}\, z\, (\overline{x}\langle z\rangle \mid !\, z.\, \overline{y'} \mid \prod_{i=1}^n W_i \mid Q') \end{aligned} \tag{66}$$

Moreover, from (64) and (66) we have

$$\boldsymbol{\nu}\, z\, (\overline{x}\langle z\rangle \mid !\, z.\, \overline{y} \mid \prod_{i=1}^n W_i \mid P') \approx \boldsymbol{\nu}\, z\, (\overline{x}\langle z\rangle \mid !\, z.\, \overline{y'} \mid \prod_{i=1}^n W_i \mid Q_7)$$

which, using Lemma 8.1 yields

$$!\, z.\, \overline{y} \mid \prod_{i=1}^n W_i \mid P' \approx !\, z.\, \overline{y'} \mid \prod_{i=1}^n W_i \mid Q_7\,. \tag{67}$$

Name $z \notin fn(\prod_{i=1}^n W_i) \cup \{y, y'\}$ and $\prod_{i=1}^n W_i$ is unidirectional; moreover, P' and $Q_7 \in \mathcal{L}[\![\mathcal{P}_{loc}]\!]$. Hence by Lemma A.3 we deduce that

$$y = y'\,. \tag{68}$$

Now we can conclude the proof of the lemma. We had already set $C[\cdot] \stackrel{def}{=} \boldsymbol{\nu}\, z\, (\overline{x}\langle z\rangle | !\, z.\, \overline{y} | [\cdot])$ and $H \stackrel{def}{=} !\, z.\, \overline{y} \mid \prod_{i=1}^n W_i$. The results in (68), (62), (65) and (67) show that the requirements in (61) are all satisfied. $\square$

Finally, the proof of Lemma 8.3, whose assertion is

Suppose $P_1, P_2 \in \mathcal{L}[\![\mathcal{P}_{loc}]\!]$. Then

$$\boldsymbol{\nu}\, z\, (\mathrm{Loc}\langle xzy\rangle \mid P_1) \approx \boldsymbol{\nu}\, z\, (\mathrm{Loc}\langle xzy'\rangle \mid P_2) \quad \text{iff} \quad y = y' \text{ and } P_1 \approx P_2.$$

Proof of Lemma 8.3

The implication from right to left follows from the congruence properties of $\approx$. We consider the implication from left to right. Set

$$\begin{array}{rrcl} & H_1 & \stackrel{def}{=} & \boldsymbol{\nu}\, z\,(\mathrm{Loc}\langle xzy\rangle \mid P_1) \\ & & \equiv & \boldsymbol{\nu}\, z\,(\overline{x}\langle z\rangle \mid !\, z.\, \overline{y} \mid P_1) \\ \\ \text{and} & H_2 & \stackrel{def}{=} & \boldsymbol{\nu}\, z\,(\mathrm{Loc}\langle xzy'\rangle \mid P_2) \\ & & \equiv & \boldsymbol{\nu}\, z\,(\overline{x}\langle z\rangle \mid !\, z.\, \overline{y'} \mid P_2) \end{array}$$

By Lemma 8.1, $H_1 \approx H_2$ implies

$$!\, z.\, \overline{y} \mid P_1 \approx !\, z.\, \overline{y'} \mid P_2\,.$$

and by Lemma A.3, we have $y = y'$. Finally, Lemma 8.2 and $!\, z.\, \overline{y} \mid P_1 \approx !\, z.\, \overline{y} \mid P_2$ imply $P_1 \approx P_2$. $\square$