Some Congruence Properties for π-calculus Bisimilarities

Michele Boreale
Università di Roma "La Sapienza"
michele@dsi.uniroma1.it

Davide Sangiorgi
INRIA, Sophia-Antipolis
davides@sophia.inria.fr

Abstract

Both for interleaving and for non-interleaving semantics, several variants of a π-calculus bisimilarity can be given which differ on the requirements imposed on name instantiations. Examples are the *late, early, open* and *ground* variants. The ground variant is the simplest because it places no requirements on name instantiations. With the exception of open bisimilarities, none of the bisimilarity considered in the literature is a congruence relation on the full π-calculus language.

We show that in the case of (certain forms of) *causal bisimulation* the late, early, open and ground variants coincide and are congruence relations in the sublanguage of the π-calculus without matching. We also show that to obtain the same results in the case of the interleaving bisimilarity, in addition to forbidding matching it is necessary to constrain the output prefix.

1 Motivations

One of the most studied and important issues in process algebra is to individuate behavioural equivalences which be pragmatically satisfactory (i.e., the identifications made on processes be sensible) and mathematically tractable (i.e., above all, process equivalences be easy to verify). For the latter point, an important property of the behavioural equivalence is *congruence*, which allows us the replacement of "equal" terms in any context.

In CCS-like process algebras, bisimulation has achieved wide consensus and popularity as a mathematical tool for defining behavioural equivalences. The simplest form of bisimulation is that of the interleaving approach. It requires that if P and Q are bisimilar, then

$$P \xrightarrow{\mu} P' \text{ implies } Q \xrightarrow{\mu} Q', \text{ for some } Q' \text{ bisimilar to } P' \qquad (*)$$

and the vice versa, on the possible transitions by Q. In a CCS transition $P \xrightarrow{\mu} P'$, action μ can be thought as the offer from P of a synchronisation with an external process.

In this paper, we deal with bisimulation-based equivalences for the *π-calculus* [9]. Intense research over the past six years has made the π-calculus *the* paradigmatic example of process algebra for *mobile* systems. Formally, π-calculus represents a development of CCS in which *communication of names* is allowed. Input and output prefixes take the form $a(\widetilde{b}).\,P$ and $\overline{a}\langle\widetilde{b}\rangle.\,P$, respectively; the former process waits for a tuple of names $\widetilde{c}$ to be sent along a and then behaves like $P\{\widetilde{c}/\widetilde{b}\}$, whereas the latter process is willing to send names $\widetilde{b}$ along a and then continues like P. Definition $(*)$ is the same in π-calculus and in CCS, up to the different syntax of actions and the fact that, in the π-calculus, identity of actions is taken modulo alpha conversion.

The noticeable feature of $(*)$ is that it requires *no* name instantiation. We call *ground bisimulation* a bisimulation with this property. Due to its simplicity, the definition itself of ground bisimulation provides us with a relatively efficient tool for checking process bisimilarities.

Unfortunately, in the π-calculus ground bisimulation is not a congruence relation. The failure is inevitable in presence of the matching operator, written $[a = b]$ and used to test for equalities between two names a and b. For instance, if a and b are different then processes

$$P \stackrel{\text{def}}{=} [a = b]\overline{b}\langle b\rangle.\,\mathbf{0} \qquad\qquad Q \stackrel{\text{def}}{=} \mathbf{0}$$

are the same since they exhibit no behaviour, but can be distinguished in the context $C[\cdot] \stackrel{\text{def}}{=} (c(a).\,[\cdot])|\overline{c}\langle b\rangle$ since $C[P]$ has a derivative which can perform an output action — the interaction between input $c(a)$ and output $\overline{c}\langle b\rangle$ sets the matching in P to true — which $C[Q]$ has not.

However, processes P and Q can be distinguished under the instantiation $\{b/a\}$, which removes the difference between names a and b. Indeed, the natural way of modifying ground bisimulation, so to get closer to a congruence relation, is to take name instantiation into account. But then, at least two serious drawbacks emerge:

1. Checking bisimilarities can become expensive, for name instantiation can cause a state explosion problem in the verification.
2. Different variants of bisimilarity are possible, in correspondence with different ways of using name instantiation. Examples are the *late*, *early* and *open* variants [9, 12].

It can be perhaps questioned whether (2) should be considered a drawback, but it is at least a source of confusion in applications. Further, the late and early variants *still fail* to be congruence relations, because not preserved by input prefix.

The above discussion suggests that it is important to isolate subcalculi of the π-calculus with a non-trivial expressiveness and forms of ground bisimulation for them which be congruence relations. For then, all mentioned forms of bisimulation coincide, and we can exploit the simplicity of ground bisimulation for proving interesting process bisimilarities. The research conducted has evidenced that, as far as expressivenesss is concerned, the operators of matching, sum and the full output prefixes play a secondary role w.r.t. the other operators (restriction, parallel composition, replication and input prefix) [6, 5, 10]. Restricted forms of output prefix, in which the continuation is null (*asynchronous output*) [6] or where all names emitted are private (*bound output*) [15], have been proposed. We are therefore interested in congruence properties for forms of ground bisimulation on subcalculi which have some syntactic limitations on matching, sum or output prefix.

We are only aware of two congruence results for ground bisimulation: (interleaving) ground bisimulation has been proved to be a congruence relation

1. in the subcalculus without matching and with only asynchronous outputs [4, 14];
2. in πI, a subcalculus of the π-calculus without matching and with only bound outputs [15].

Both results are obtained by restraining the output construct. They do not show, however, the necessity of these limitations. For instance, one might hope to achieve the same results by forbidding summation but retaining the full output prefix. This paper contains two main contributions:

1. We show that if the π-calculus language includes the full output prefix and has a non-trivial expressiveness (i.e., it includes constructs for parallelism, replication and restriction) then ground bisimulation is not a congruence relation, neither in the strong nor in the weak case.
2. We show that the full output prefix is tolerated if ground bisimulation is strengthened so to reveal certain *causal* dependencies among actions, namely those which originate from the nesting of prefixes and which are propagated through interactions [7, 3]. Both *strong* and *weak* forms of *ground causal bisimulation* are congruence relations in absence of matching.

The two points are developed in Sections 3 and 4. In Section 2 the π-calculus and some basic notions on interleaving semantics are presented.

2 Background

2.1 The π-calculus

The countable set $\mathcal{N}$ of *names* is ranged over by $a, b, \ldots, x, y, \ldots$. *Processes* are ranged over by P, Q and R. The π-calculus syntax we shall work with is built from the operators of guarded summation, restriction, parallel composition and replication:

$$\begin{aligned} P \quad &:= \textstyle\sum_{i \in I} \alpha_i . P_i \quad | \quad \boldsymbol{\nu}\, a\, P \quad | \quad P_1 \,|\, P_2 \quad | \quad !\, P \\ \alpha \quad &:= a(\widetilde{b}) \quad | \quad \overline{a}\langle\widetilde{b}\rangle \quad | \quad \tau\,. \end{aligned}$$

$$
\begin{array}{llll}
\texttt{Sum}: & \sum_{i\in I}\alpha_i.P_i \xrightarrow{\alpha_j} P_j,\ j\in I & \texttt{Rep}: & \dfrac{P \mid !\,P \xrightarrow{\mu} P'}{!\,P \xrightarrow{\mu} P'} \\[3ex]
\texttt{Par}: & \dfrac{P_1 \xrightarrow{\mu} P_1'}{P_1 \mid P_2 \xrightarrow{\mu} P_1' \mid P_2} & \texttt{Com}: & \dfrac{P_1 \xrightarrow{(\boldsymbol{\nu}\,\widetilde{b'})\overline{a}\langle\widetilde{b}\rangle} P_1' \qquad P_2 \xrightarrow{a(\widetilde{c})} P_2'}{P_1 \mid P_2 \xrightarrow{\tau} \boldsymbol{\nu}\,\widetilde{b'}\,(P_1' \mid P_2'\{\widetilde{b}/\widetilde{c}\})} \\[3ex]
\texttt{Res}: & \dfrac{P \xrightarrow{\mu} P'}{\boldsymbol{\nu}\,c\,P \xrightarrow{\mu} \boldsymbol{\nu}\,c\,P'},\ c\notin \mathrm{n}(\mu) & \texttt{Open}: & \dfrac{P \xrightarrow{(\boldsymbol{\nu}\,\widetilde{b'})\overline{a}\langle\widetilde{b}\rangle} P'}{\boldsymbol{\nu}\,c\,P \xrightarrow{(\boldsymbol{\nu}\,\widetilde{b'}c)\overline{a}\langle\widetilde{b}\rangle} P'},\ c\neq a, c\in\widetilde{b}-\widetilde{b'}
\end{array}
$$

Table 1: Interleaving operational semantics for $\mathcal{P}$.

The prefixes $a(\widetilde{b})$ and $\overline{a}\langle\widetilde{b}\rangle$ are called, respectively, input and output prefix; in the input prefix $a(\widetilde{b})$, the components of $\widetilde{b}$ are pairwise distinct. In summations, the index-set I is finite; for $\sum_{i\in\emptyset}\alpha_i.P_i$ the symbol $\mathbf{0}$ is also used, while binary summation $\sum_{i\in\{1,2\}} P_i$ is often written as $P_1 + P_2$. We will write $a.\,P$ and $\overline{a}.\,P$ when no name is carried by a. We abbreviate $\alpha.\,\mathbf{0}$ as α and $\boldsymbol{\nu}\,a\,\boldsymbol{\nu}\,b\,P$ as $(\boldsymbol{\nu}\,a,b)P$.

W.r.t. the syntax in [9] we have omitted the matching construct, for the reasons explained in the introduction, and we only admitd *guarded* summation since, by contrast with full summation, it preserves bisimilarity even in the weak case, that is when silent actions are partially ignored in the bisimilarity clauses. We have chosen to present our work in the polyadic π-calculus (where tuples of names can be passed in communications), as opposed to the monadic calculus (where exactly one name can be passed) because it makes some of the technical material simpler (for instance the counterexamples of Section 3 become harder to read, when coded up in the monadic calculus).

Input prefix $a(\widetilde{b})$ and restriction $\boldsymbol{\nu}\,a$ act as *binders* for names $\widetilde{b}$ and a, respectively. *Free names*, *bound names* of a process P, written $\mathrm{fn}(P)$ and $\mathrm{bn}(P)$ respectively, arise as expected; the *names* of P, written $\mathrm{n}(P)$ are $\mathrm{fn}(P)\cup\mathrm{bn}(P)$. *Substitutions*, ranged over by $\sigma,\sigma'\ldots$ are functions from $\mathcal{N}$ to $\mathcal{N}$; for any expression E, we write $E\sigma$ for the expression obtained from applying σ to E. Composition of two substitutions σ and σ' is written $\sigma\sigma'$. We assume the following decreasing order of precedence when writing process expressions: substitution, prefix, replication, restriction, parallel composition, binary summation.

The transition rules for the π-calculus operators are given in Table 1. *Actions*, ranged over by μ, can be of three forms: τ (interaction), $a(\widetilde{b})$ (input), or $\boldsymbol{\nu}\,\widetilde{b'}\,\overline{a}\langle\widetilde{b}\rangle$ (output). Functions bn(), fn() and n() are extended to actions as expected, once we set $\mathrm{bn}(a(\widetilde{b}))=\widetilde{b}$ and $\mathrm{bn}(\boldsymbol{\nu}\,\widetilde{b'}\,\overline{a}\langle\widetilde{b}\rangle)=\widetilde{b'}$.

Throughout the paper, we work up to α-conversion on names — that is, we implicitly take an underlying representation of names based on de Bruijn indices [1] — so to avoid tediouos side conditions in transition rules and bisimulation clauses. Therefore, for instance, in a process bound names are assumed different from each other and from the free names, and α-equivalent processes are assumed to have the same transitions. All our notations are extended to tuples componentwise.

Following Milner [8], we only admit *well-sorted processes*, that is processes which obey a predefined *sorting* discipline in their manipulation of names. The sorting prevents arity mismatching in communications, like in $\overline{a}\langle b,c\rangle.\,P \mid a(x).\,Q$. Moreover, substitutions must map names onto names of the same sort. We do not present the sorting system because it is not essential to understand the contents of this paper.

We call:

- $\mathcal{P}$ the above set of π-calculus processes;
- $\mathcal{P}^{\mathrm{a}}$ the subset of $\mathcal{P}$ in which an output prefix has no continuation, i.e., outputs are of the form $\overline{a}\langle b\rangle.\,\mathbf{0}$; we call this form of prefixing *asynchronous output*.
- $\mathcal{P}^{-}$ the subset of $\mathcal{P}$ without summation.

2.2 Bisimulations

A few forms of (interleaving) bisimilarity have been proposed for the π-calculus, notably the *late, early* and *open* bisimilarities [9, 12]. We only recall the definition of early bisimilarity.

Definition 2.1 (strong early bisimilarity) *A symmetric relation* $\mathcal{R} \subseteq \mathcal{P} \times \mathcal{P}$ *is a* strong early bisimulation *if* $P \, \mathcal{R} \, Q$ *implies:*

1. *whenever* $P \xrightarrow{a(\widetilde{b})} P'$ *for all names* $\widetilde{c}$ *there exists* Q' *s.t.* $Q \xrightarrow{a(\widetilde{b})} Q'$ *and* $P'\{\widetilde{c}/\widetilde{b}\} \, \mathcal{R} \, Q'\{\widetilde{c}/\widetilde{b}\}$*;*
2. *whenever* $P \xrightarrow{\mu} P'$ *and* μ *is not an input action, there exists* Q' *s.t.* $Q \xrightarrow{\mu} Q'$ *and* $P' \, \mathcal{R} \, Q'$*.*

Two processes P *and* Q *are* strongly early bisimilar*, written* $P \sim_{\mathrm{e}} Q$*, if* $P \, \mathcal{R} \, Q$ *for some strong early bisimulation* $\mathcal{R}$*.*

Late bisimilarity, written $\sim_{\mathrm{la}}$, inverts the order of the existential and universal quantifiers in the input clause thus:

$$\text{If } P \xrightarrow{a(\widetilde{b})} P', \text{ then there exists } Q' \text{ s.t. } Q \xrightarrow{a(\widetilde{b})} Q' \text{ and for all } \widetilde{c}, \ P'\{\widetilde{c}/\widetilde{b}\} \, \mathcal{R} \, Q'\{\widetilde{c}/\widetilde{b}\}.$$

Late bisimilarity is strictly included in early bisimilarity. *Open bisimilarity* is a stronger equality than the late and early ones. In open bisimilarity, substitutions are used in a global fashion, requiring that the bisimilarity relation itself be closed under substitutions. Moreover, the mechanism of *distinction* is used to record the fact that a restricted name cannot be identified with other names. In the language $\mathcal{P}$, open bisimulation is a congruence relation whereas late and early bisimulation are not because they are not preserved by input prefix [9].

The simplest form of bisimulation is the one where no name instantiation at all appears, apart from α-conversion. We call it *ground bisimulation.*

Definition 2.2 (strong ground bisimilarity) *A symmetric relation* $\mathcal{R} \subseteq \mathcal{P} \times \mathcal{P}$ *is a* strong ground bisimulation *if* $P \, \mathcal{R} \, Q$ *and* $P \xrightarrow{\mu} P'$ *imply that there exists* Q' *s.t.* $Q \xrightarrow{\mu} Q'$ *and* $P' \, \mathcal{R} \, Q'$*. Two processes* P *and* Q *are* strongly ground bisimilar*, written* $P \sim_{\mathrm{g}} Q$*, if* $P \, \mathcal{R} \, Q$ *for some strong bisimulation* $\mathcal{R}$*.*

The weak versions of the bisimulations, where one ignores silent steps in matching transitions, are obtained in the standard way. Let $\Longrightarrow$ be the reflexive and transitive closure of $\xrightarrow{\tau}$, let $\stackrel{\mu}{\Longrightarrow}$ be $\Longrightarrow \xrightarrow{\mu} \Longrightarrow$, and let $P \stackrel{\widehat{\mu}}{\Longrightarrow} Q$ be $P \stackrel{\mu}{\Longrightarrow} Q$, if $\mu \neq \tau$, and $P \Longrightarrow Q$, if $\mu = \tau$. Then weak ground bisimilarity is defined by replacing in Definition 2.2 the transition $Q \xrightarrow{\mu} Q'$ with $Q \stackrel{\widehat{\mu}}{\Longrightarrow} Q'$. Similarly one defines weak open bisimulation. In the definitions of weak late and early bisimulation the input clause is a little different because the name instantiation must occur immediately after the input is performed. The input clause for weak early bisimulation is:

$$\begin{array}{c}\text{If } P \xrightarrow{a(\widetilde{b})} P', \text{ then for all } \widetilde{c} \text{ there exists } Q' \text{ s.t. } Q \Longrightarrow \xrightarrow{a(\widetilde{b})} Q' \text{ and,} \\ \text{for some } Q'', \ Q'\{\widetilde{c}/\widetilde{b}\} \Longrightarrow Q'' \text{ and } P'\{\widetilde{c}/\widetilde{b}\} \, \mathcal{R} \, Q''.\end{array}$$

The input clause for weak late bisimulation is:

$$\begin{array}{c}\text{If } P \xrightarrow{a(\widetilde{b})} P', \text{ then there exists } Q' \text{ s.t. } Q \Longrightarrow \xrightarrow{a(\widetilde{b})} Q' \text{ and for all } \widetilde{c} \\ \text{there is } Q'' \text{ s.t. } Q'\{\widetilde{c}/\widetilde{b}\} \Longrightarrow Q'' \text{ and } P'\{\widetilde{c}/\widetilde{b}\} \, \mathcal{R} \, Q''.\end{array}$$

We use the symbols $\approx_{\mathrm{g}}, \approx_{\mathrm{la}}, \approx_{\mathrm{e}}$ and $\approx_{\mathrm{o}}$ for the weak versions of ground, late, early and open bisimulation, respectively.

It was proved, independently in [4] and [14], developing an earlier result by Honda [5], that in absence of matching and of continuation underneath the output prefix, ground bisimulation is a congruence.[1] A key lemma for proving the congruence of ground bisimulation is its closure under substitutions.

Lemma 2.3 *Relations* $\sim_{\mathrm{g}}$ *and* $\approx_{\mathrm{g}}$ *are preserved by name instantiations in the language* $\mathcal{P}^{\mathrm{a}}$*.*

Theorem 2.4 $\sim_{\mathrm{g}}$ *and* $\approx_{\mathrm{g}}$ *are congruence relations in the language* $\mathcal{P}^{\mathrm{a}}$*.*

[1] The language in [14] does not have summation; it is straightoforward to accommodate guarded summations in the proof.

Corollary 2.5

1. *Relations* $\sim_{\rm g}, \sim_{\rm la}, \sim_{\rm e}, \sim_{\rm o}$ *coincide in the language* $\mathcal{P}^{\rm a}$*;*

2. *Relations* $\approx_{\rm g}, \approx_{\rm la}, \approx_{\rm e}, \approx_{\rm o}$ *coincide in the language* $\mathcal{P}^{\rm a}$*.*

Proof: For (1), the inclusions $\sim_{\rm o} \subseteq \sim_{\rm la} \subseteq \sim_{\rm e} \subseteq \sim_{\rm g}$ follow directly from the definitions. Using the fact that $\sim_{\rm g}$ is closed under substitutions one can prove $\sim_{\rm g} \subseteq \sim_{\rm o}$.
Assertion (2) is proved similarly. $\Box$

3 Non-congruence results

We show that in Theorem 2.4 and Corollary 2.5 the limitation to asynchronous outputs is important. The results fail if the language includes, in addition to full output prefix, at least the operators of parallelism, replication, restriction and input prefix. In the counterexamples below, syntactically different letters stand for diffent names.

It was known [9] that ground bisimulation is not preserved by name substitutions in presence of the matching construct (see introductory section), or in the language $\mathcal{P}$, with both full output prefix and summation, as the following counterexample shows.

Counterexample 3.1 (from [9], language $\mathcal{P}$) *Let* $P \stackrel{\rm def}{=} \overline{x} \mid y$ *and* $Q \stackrel{\rm def}{=} \overline{x}.\, y + y.\, \overline{x}$*. Then* $P \sim_{\rm g} Q$*, but* $P\{x/y\} \not\sim_{\rm g} Q\{x/y\}$*, since* $P\{x/y\}$ *can terminate without performing visible actions.*

The same pair of processes show that neither $\approx_{\rm g}$ *is preserved by substitutions in* $\mathcal{P}$*.* $\Box$

The problem is more interesting in the language $\mathcal{P}^-$, where summation is forbidden. Indeed, Counterexample 3.1 is based on the expansion law, which makes no sense without summation. We show that even in $\mathcal{P}^-$ ground bisimulation is not preserved by name substitutions, neither in the strong nor in the weak case. First, some simple laws.

Lemma 3.2

1. $\boldsymbol{\nu}\, b\, (\overline{z}\langle d\rangle .\, \overline{b}.\, P \mid x(e).\, b.\, Q) \sim_{\rm la} \overline{z}\langle d\rangle .\, x(e).\, \tau.\, \boldsymbol{\nu}\, b\, (P \mid Q) + x(e).\, \overline{z}\langle d\rangle .\, \tau.\, \boldsymbol{\nu}\, b\, (P \mid Q)$,
   if $e \notin {\rm fn}(\overline{z}\langle d\rangle .\, \overline{b}.\, P)$*,* $b \notin \{z, d, x, e\}$ *and* $z \neq x$*.*

2. $!\, \boldsymbol{\nu}\, \widetilde{d}\, (P + Q) \sim_{\rm la} !\, \boldsymbol{\nu}\, \widetilde{d}\, P \mid \, !\, \boldsymbol{\nu}\, \widetilde{d}\, Q$

3. $\alpha.\, \tau.\, P \approx_{\rm la} \alpha.\, P$*.*

Proof: (1) is a simple form of the expansion law; (2) is taken from [13]; (3) is one of the ordinary τ-laws. $\Box$

Counterexample 3.3 (language $\mathcal{P}^-$, strong case) *The following counterexample shows that* $\sim_{\rm g}$ *is not preserved by name substitutions in* $\mathcal{P}^-$*. Take*

$$P \stackrel{\rm def}{=} \; !\, \overline{z}.\, x.\, \tau.\, y \mid \, !\, x.\, \overline{z}.\, \tau.\, y \qquad\qquad Q \stackrel{\rm def}{=} \; !\, \boldsymbol{\nu}\, b\, (\overline{z}.\, \overline{b} \mid x.\, b.\, y)$$

Using the law (1) in Lemma 3.2 and some garbage collection of restrictions, we get

$$\boldsymbol{\nu}\, b\, (\overline{z}.\, \overline{b} \mid x.\, b.\, y) \sim_{\rm la} \overline{z}.\, x.\, \tau.\, y + x.\, \overline{z}.\, \tau.\, y$$

Therefore, since $\sim_{\rm la}$ *is preserved by replication and is contained in* $\sim_{\rm g}$*, and using law (2) of Lemma 3.2, we obtain:*

$$\begin{array}{rcl} Q & \sim_{\rm g} & !\, \big(\overline{z}.\, x.\, \tau.\, y + x.\, \overline{z}.\, \tau.\, y\big) \\ & \sim_{\rm g} & !\, \overline{z}.\, x.\, \tau.\, y \mid \, !\, x.\, \overline{z}.\, \tau.\, y \quad = \quad P \end{array}$$

However, $P\{z/x\} \not\sim_{\rm g} Q\{z/x\}$*, since* $Q\{z/x\}$ *can perform a move at* y *after two steps, whereas* $P\{z/x\}$ *only after three steps.* $\Box$

Counterexample 3.4 (language $\mathcal{P}^-$, weak case) *The following counterexample shows that $\approx_{\mathrm{g}}$ is not preserved by name substitutions in $\mathcal{P}^-$. We write $\overline{a}(h).\, R$ as abbreviation for $(\boldsymbol{\nu}\, h\,)\overline{a}\langle h\rangle.\, R$.*

Take

$$
\begin{aligned}
P &\stackrel{\text{def}}{=} \; !\,\boldsymbol{\nu}\, d\, \Big(\overline{z}\langle d\rangle.\, x(e).\, \overline{a}(h).\, \overline{h}\langle d\rangle.\, \overline{h}\langle e\rangle\Big) \mid \, !\,\boldsymbol{\nu}\, d\, \Big(x(e).\, \overline{z}\langle d\rangle.\, \overline{a}(h).\, \overline{h}\langle d\rangle.\, \overline{h}\langle e\rangle\Big)\\
Q &\stackrel{\text{def}}{=} \; !\,(\boldsymbol{\nu}\, d, b\,)\Big(\overline{z}\langle d\rangle.\, \overline{b} \mid x(e).\, b.\, \overline{a}(h).\, \overline{h}\langle d\rangle.\, \overline{h}\langle e\rangle\Big)
\end{aligned}
$$

Proceeding as in the previous example and, in addition, using law (3) of Lemma 3.2, we obtain:

$$
\begin{aligned}
Q \;&\sim_{\mathrm{g}}\; !\,\boldsymbol{\nu}\, d\, \Big(\overline{z}\langle d\rangle.\, x(e).\, \tau.\, \overline{a}(h).\, \overline{h}\langle d\rangle.\, \overline{h}\langle e\rangle + x(e).\, \overline{z}\langle d\rangle.\, \tau.\, \overline{a}(h).\, \overline{h}\langle d\rangle.\, \overline{h}\langle e\rangle\Big)\\
&\approx_{\mathrm{g}}\; !\,\boldsymbol{\nu}\, d\, \Big(\overline{z}\langle d\rangle.\, x(e).\, \overline{a}(h).\, \overline{h}\langle d\rangle.\, \overline{h}\langle e\rangle + x(e).\, \overline{z}\langle d\rangle.\, \overline{a}(h).\, \overline{h}\langle d\rangle.\, \overline{h}\langle e\rangle\Big)\\
&\sim_{\mathrm{g}}\; !\,\boldsymbol{\nu}\, d\, \big(\overline{z}\langle d\rangle.\, x(e).\, \overline{a}(h).\, \overline{h}\langle d\rangle.\, \overline{h}\langle e\rangle\big) \mid \, !\,\boldsymbol{\nu}\, d\, \big(x(e).\, \overline{z}\langle d\rangle.\, \overline{a}(h).\, \overline{h}\langle d\rangle.\, \overline{h}\langle e\rangle\big) \;=\; P
\end{aligned}
$$

which proves $P \approx_{\mathrm{g}} Q$. But $P\{z/x\}$ and $Q\{z/x\}$ are not in the relation $\approx_{\mathrm{g}}$. Consider the sequence of actions from $Q\{z/x\}$:

$$
\begin{aligned}
Q\{z/x\} &\xrightarrow{\;\tau\;} (\boldsymbol{\nu}\, b, d\,)\big(\overline{b} \mid b.\, \overline{a}(h).\, \overline{h}\langle d\rangle.\, \overline{h}\langle d\rangle.\,\big) \mid Q\{z/x\}\\
&\xrightarrow{\;\tau\;} \boldsymbol{\nu}\, d\, \big(\overline{a}(h).\, \overline{h}\langle d\rangle.\, \overline{h}\langle d\rangle.\,\big) \mid Q\{z/x\}\\
&\xrightarrow{\overline{a}(h)} \xrightarrow{\overline{h}\langle d\rangle} \xrightarrow{\overline{h}\langle d\rangle} Q\{z/x\}.
\end{aligned}
$$

There are two consecutive actions at h which carry the same name. This behaviour is not possible for $P\{z/x\}$ since, in any subcomponent $\overline{a}(h).\, \overline{h}\langle d\rangle.\, \overline{h}\langle e\rangle$ of $P\{z/x\}$ name e cannot be instantiated with name d. Therefore $P\{z/x\}$ cannot match the above sequence of transitions from $Q\{z/x\}$ and hence $P\{z/x\} \not\approx_{\mathrm{g}} Q\{z/x\}$. □

4 Causal bisimulations

To obtain a form of ground bisimulation which is a congruence relation on the language $\mathcal{P}$, we have to abandon interleaving bisimilarity and move to *non-interleaving* bisimilarities, more precisely to those which take *causality* into account.

Causal dependencies induced by action prefix (e.g. the fact that in $\alpha.\beta$ the execution of β is enabled by the execution of α) and propagated through communications are not revealed by the interleaving transition system. In order to take such dependencies into account, we adopt a form of operational semantics with explicit causes, following Kiehn's approach for CCS [7], adapted to the π-calculus in [2].

For the extra causal information, we use an auxiliary set $\mathcal{K}$ of *causes.* In the enriched system, visible transitions are of the form $A \xrightarrow[K,k]{\mu} A'$. The visible action μ is associated with a unique cause $k \in \mathcal{K}$. The set of actions from which μ is causally related (its set of causes) is revealed by means of the set $K \subseteq_{\mathrm{fin}} \mathcal{K}$ of their associated causes. If a successive action μ' is caused by μ, then μ' will have k (i.e., the cause associated with μ) in its set of causes. In the term syntax, the explicit use of causes is accompanied by the introduction of a causal prefix $K :: A$, which says that the set of causes of any action of the process A must contain K.

By contrast, silent actions are not observable and do not exhibit causes. However, causes do play an important role in the communication rule, for the causes of two interacting actions must be appropriately merged. Consider the sequence of transitions

$$
\begin{aligned}
&\boldsymbol{\nu}\, b\, \big(a.\, b.\, c.\, \mathbf{0} \mid d.\, \overline{b}.\, \mathbf{0}\big) \xrightarrow[\emptyset\,, k_1]{a} \boldsymbol{\nu}\, b\, \big(\{k_1\} :: b.\, c.\, \mathbf{0} \mid d.\, \overline{b}.\, \mathbf{0}\big) \xrightarrow[\emptyset\,, k_2]{d} \boldsymbol{\nu}\, b\, \big(\{k_1\} :: b.\, c.\, \mathbf{0} \mid \{k_2\} :: \overline{b}.\, \mathbf{0}\big)\\
&\xrightarrow{\;\tau\;} \boldsymbol{\nu}\, b\, \big(\{k_1, k_2\} :: c.\, \mathbf{0} \mid \{k_1, k_2\} :: \mathbf{0}\big) \xrightarrow[\{k_1, k_2\}, k_3]{c} \boldsymbol{\nu}\, b\, \big(\{k_1, k_2\} :: \{k_3\} :: \mathbf{0} \mid \{k_1, k_2\} :: \mathbf{0}\big)\,.
\end{aligned}
$$

They show that the actions at a and d have no cause; and that the action at c is causally dependent on a and d. The latter makes sense: c could fire only because both a and d fired. Note that in the τ-transition, the sets of causes $\{k_1\}$ and $\{k_2\}$ of the consumed actions are merged.

$\mathcal{K}$ is the infinite set of *causes*; k and h range over causes; K and H over finite subsets of $\mathcal{K}$. The sets $\mathcal{K}$ of causes and $\mathcal{N}$ of names are disjoint. The language of *causal processes*, written $\mathcal{P}_{\mathrm{c}}$ and ranged over by $A, B, \ldots$, is given by the following grammar:

$$A := P \;\mid\; K :: A \;\mid\; A \mid A \;\mid\; \boldsymbol{\nu}\, a\, A$$

$$\texttt{Sum}: \ \sum_{i\in I}\alpha_i.P_i \xrightarrow[\emptyset\,,k]{\alpha_j} k :: P_j,\ j\in I \qquad \texttt{Rep}: \ \frac{P \mid !\,P \xrightarrow[K,k]{\mu} P'}{!\,P \xrightarrow[K,k]{\mu} P'}$$

$$\texttt{Cau}: \ \frac{A \xrightarrow[K,k]{\mu} A'}{K' :: A \xrightarrow[K\cup K',k]{\mu} K' :: A'} \qquad \texttt{Par}: \ \frac{A_1 \xrightarrow[K,k]{\mu} A_1'}{A_1 \mid A_2 \xrightarrow[K,k]{\mu} A_1' \mid A_2}$$

$$\texttt{Res}: \ \frac{A \xrightarrow[K,k]{\mu} A'}{\boldsymbol{\nu}\, c\, A \xrightarrow[K,k]{\mu} \boldsymbol{\nu}\, c\, A'},\ c\notin \mathrm{n}(\mu) \qquad \texttt{Open}: \ \frac{A \xrightarrow[K,k]{(\boldsymbol{\nu}\,\widetilde{b'})\overline{a}\langle\widetilde{b}\rangle} A'}{\boldsymbol{\nu}\, c\, A \xrightarrow[K,k]{(\boldsymbol{\nu}\,\widetilde{b'}c)\overline{a}\langle\widetilde{b}\rangle} A'},\ c\neq a, c\in \widetilde{b}-\widetilde{b'}$$

$$\texttt{T-par}: \ \frac{A_1 \xrightarrow{\tau} A_1'}{A_1 \mid A_2 \xrightarrow{\tau} A_1' \mid A_2} \qquad \texttt{T-res}: \ \frac{A \xrightarrow{\tau} A'}{\boldsymbol{\nu}\, c\, A \xrightarrow{\tau} \boldsymbol{\nu}\, c\, A'}$$

$$\texttt{Com}: \ \frac{A_1 \xrightarrow[K_1,k]{(\boldsymbol{\nu}\,\widetilde{b'})\overline{a}\langle\widetilde{b}\rangle} A_1' \qquad A_2 \xrightarrow[K_2,k]{a(\widetilde{c})} A_2'}{A_1 \mid A_2 \xrightarrow{\tau} \boldsymbol{\nu}\,\widetilde{b'}\,(A_1'[k \rightsquigarrow K_2] \mid A_2'\{\widetilde{b}/\widetilde{c}\}[k\rightsquigarrow K_1])}\ k\notin \mathcal{K}(A_1,A_2)$$

$$\texttt{T-cau}: \ \frac{A \xrightarrow{\tau} A'}{K :: A \xrightarrow{\tau} K :: A'} \qquad \texttt{T-rep}: \ \frac{A \mid !\,A \xrightarrow{\tau} A'}{!\,A \xrightarrow{\tau} A'}$$

Table 2: Transition rules for visible and silent actions of causal processes.

where P is a standard $\mathcal{P}$-process, as defined in Section 2. The above syntax does not allow the presence of causes underneath dynamic operators (prefixes, sums and replications), because we are only interested in derivatives of standard processes, for which these cases may never arise.

The rule for operational semantics of $\mathcal{P}_c$ are reported in Table 2. A *cause substitution* $[k \rightsquigarrow K]$ denotes the substitution of the cause k with the set K; e.g., $\big(\{k_1,k_2\} :: a(\widetilde{b}).\mathbf{0}\big)[k_1 \rightsquigarrow \{k_3,k_4\}]$ is $\{k_3,k_4,k_2\} :: a(\widetilde{b}).\mathbf{0}$. Union of causes is often denoted by a comma; e.g. $K\cup\{k\}$ is denoted by (K,k). We abbreviate $\{k\} :: A$ as $k :: A$. Causal prefix $K :: A$ has the same precedence as ordinary prefix. We say that a cause k is *fresh* for a causal process A if k does not appear in A.

We are now set to introduce *causal bisimilarity* [3, 7, 2]. As we did in Section 2, we only present the early and ground variants. Late and open variants are defined in the expected way. Our main results will be that both in the strong and in the weak cases, in the language $\mathcal{P}$ all variants of causal bisimilarity are congruence relations and coincide with each other. The counterexamples of Section 3 do not work anymore because causal bisimulation allows us to detect that certain output and input actions are produced by prefixes that are sequentially ordered and hence, no matter what substitutions are applied to the processes, the prefixes can never combine into a communication. For instance, the processes $Q \stackrel{\mathrm{def}}{=} \overline{x}.\,y$ in Counterexample 3.1 has transitions

$$Q \xrightarrow[\emptyset\,,k_1]{\overline{x}} \{k_1\} :: y \xrightarrow[k_1,k_2]{y} \{\{k_1\},k_2\} :: \mathbf{0}$$

The occurrence of k_1 in the second transition shows a dependency with the first transition. By contrast, the only transitions from process $P \stackrel{\mathrm{def}}{=} \overline{x} \mid y$ with labels $\overline{x}$ and y are:

$$P \xrightarrow[\emptyset\,,k_1]{\overline{x}} \{k_1\} :: \mathbf{0} \mid y \xrightarrow[\emptyset\,,k_2]{y} \{k_1\} :: \mathbf{0} \mid \{k_2\} :: \mathbf{0}$$

The causality information shows that these two transitions of P are independent. As a consequence, processes P and Q are *not* causally bisimilar.

We only report the proofs for the weak case, which is more difficult. As usual, the "weak causal arrow" $A \overset{\mu}{\underset{K,k}{\Longrightarrow}} A'$ stands for $A \Longrightarrow \xrightarrow[K,k]{\mu} \Longrightarrow A'$.

Definition 4.1 (weak early causal bisimilarity) *A binary symmetric relation $\mathcal{R}$ over causal processes is an* early causal bisimulation *if, whenever $A \mathrel{\mathcal{R}} B$, then:*

- *whenever* $A \xrightarrow{\tau} A'$ *then* $B \Longrightarrow B'$, *for some* B' *s.t.* $A' \mathcal{R} B'$, *and*
- *whenever* $A \xrightarrow[K,k]{\mu} A'$, *with* k *fresh for* A *and* B*:*
  1. *if* $\mu = a(\widetilde{b})$ *then for all* $\widetilde{c}$ *there exists* B' *s.t.* $B \Longrightarrow\xrightarrow[K,k]{\mu} B'$ *and, for some* B'', $B'\{\widetilde{c}/\widetilde{b}\} \Longrightarrow B''$ *and* $A'\{\widetilde{c}/\widetilde{b}\} \mathcal{R} B''$*;*
  2. *if* μ *is not an input action then there exists* B' *s.t.* $B \overset{\mu}{\underset{K,k}{\Longrightarrow}} B'$ *and* $A' \mathcal{R} B'$*.*

A and B are early causal bisimilar, *written* $A \approx^{\mathrm{c}}_{\mathrm{e}} B$*, if* $A \mathcal{R} B$ *for an early causal bisimulation* $\mathcal{R}$*.*

Definition 4.2 (weak ground causal bisimilarity) *A binary symmetric relation* $\mathcal{R}$ *over causal processes is a* ground causal bisimulation *if, whenever* $A \mathcal{R} B$*, then:*

- *whenever* $A \xrightarrow{\tau} A'$ *then there exists* B' *s.t.* $B \Longrightarrow B'$ *and* $A' \mathcal{R} B'$*, and*
- *whenever* $A \xrightarrow[K,k]{\mu} A'$*, with* k *fresh for* A *and* B*, then there exists* B' *s.t.* $B \overset{\mu}{\underset{K,k}{\Longrightarrow}} B'$ *and* $A' \mathcal{R} B'$*.*

A and B are ground causal bisimilar*, written* $A \approx^{\mathrm{c}}_{\mathrm{g}} B$*, if* $A \mathcal{R} B$ *for an ground causal bisimulation* $\mathcal{R}$*.*

4.1 Auxiliary lemmas

In this section we establish a few results on operational semantics and ground causal bisimulation, which are used in subsection 4.2 to prove the main results. In some of our proofs, it will be convenient to use a *causal structural congruence* relation. It is the natural extension to causal processes of Milner's structural congruence for standard processes [8]. More precisely, we let causal structural congruence be the least congruence over causal processes generated by the following axioms:

1. $A|B \equiv A|B, \quad A|(B|C) \equiv (A|B)|C, \quad A|\mathbf{0} \equiv A$;
2. $\boldsymbol{\nu}\, a\, \mathbf{0} \equiv \mathbf{0}, \quad \boldsymbol{\nu}\, a\, \boldsymbol{\nu}\, b\, A \equiv \boldsymbol{\nu}\, b\, \boldsymbol{\nu}\, a\, A$;
3. $(\boldsymbol{\nu}\, a\, A)|B \equiv \boldsymbol{\nu}\, a\, (A|B)$, if a not free in B;
4. $!\,P \equiv P|\,!\,P$;
5. $\emptyset :: A \equiv A, \quad K_1 :: K_2 :: A \equiv K_1 \cup K_2 :: A$;
6. $K :: (A_1|A_2) \equiv (K :: A_1)|(K :: A_2), \quad K :: \boldsymbol{\nu}\, \widetilde{c}\, A \equiv \boldsymbol{\nu}\, \widetilde{c}\, K :: A$.

Lemma 4.3 $\equiv$ *is a strong ground causal bisimulation and is preserved by substitutions.*

In the sequel, we will freely apply Lemma 4.3 without recalling it. We use $A \Longrightarrow\equiv A'$ to mean that there is a A'' s.t. $A \Longrightarrow A''$ and $A'' \equiv A'$. The next lemma, stating that both relations $\Longrightarrow$ and $\approx^{\mathrm{c}}_{\mathrm{g}}$ are preserved by cause substitution, is proved in [2] (see Lemmas 4.6(3) and 4.10; the latter proves it for $\approx^{\mathrm{c}}_{\mathrm{e}}$, but the proof for $\approx^{\mathrm{c}}_{\mathrm{g}}$ is the same and consists in defining a simple bisimulation relation).

Lemma 4.4 *Let* A*,* B *be causal processes and let* ρ *be a cause substitution.*

1. $A \Longrightarrow A'$ *implies* $A\rho \Longrightarrow A'\rho$*;*
2. $A \approx^{\mathrm{c}}_{\mathrm{g}} B$ *implies* $A\rho \approx^{\mathrm{c}}_{\mathrm{g}} B\rho$*.*

When a transition $A \xrightarrow[K,k]{\mu} A'$, with k fresh, takes place, inside A' name k acts as a "pointer" to the location which originates the action. Lemma 4.5 reveals the structural relationship between source and target terms of such a transition. In the assertion of this lemma, P represents the location where the action comes from, and B represent "the rest of the process".

Lemma 4.5 (structural lemma) *Let A be a causal process and suppose that $A \xrightarrow[K,k]{\mu} A'$ with k fresh for A. Then there are $\widetilde{c}, \widetilde{b}', \alpha$, P, Q and B with k fresh for B s.t.:*

a) $A \equiv (\boldsymbol{\nu}\,\widetilde{c}, \widetilde{b}')(K :: (Q + \alpha.P) \mid B)$ *and, if μ is an input action then $\alpha = \mu$ and $\widetilde{b}' = \emptyset$; if μ is an output action, then $\mu = \boldsymbol{\nu}\,\widetilde{b}'\,\overline{a}\langle\widetilde{b}\rangle$ and $\alpha = \overline{a}\langle\widetilde{b}\rangle$, for some a and $\widetilde{b}$.*

b) $A' \equiv \boldsymbol{\nu}\,\widetilde{c}\,((K,k) :: P \mid B)$.

PROOF: A simple transition induction on $A \xrightarrow[K,k]{\mu} A'$. $\square$

Lemma 4.6 relates the transitions of A to those of $A\sigma$, for any process A and substitution σ. Part 4 of the lemma shows that each τ-move from $A\sigma$ either corresponds to a τ-move from A or it can be decomposed into two *independent* complementary transitions from A. The independence is given by the fact that the cause name associated to the first transition (k_1) does not occur in the set of causes of the second one (K_2). In the assertion of the lemma, recall that, by our convention on bound names, we can assume that substitutions do not touch bound names of terms and actions.

Lemma 4.6 (correspondence between $A\sigma$ and A) *Let σ be a substitution and A be a causal process.*

1. $A \xrightarrow[K,k]{\mu} A'$ *(resp.* $A \xrightarrow{\tau} A'$ *) implies* $A\sigma \xrightarrow[K,k]{\mu\sigma} A'\sigma$ *(resp.* $A\sigma \xrightarrow{\tau} A'\sigma$ *).*

2. $A \stackrel{\mu}{\underset{K,k}{\Longrightarrow}} A'$ *(resp.* $A \Longrightarrow A'$ *) implies* $A\sigma \stackrel{\mu\sigma}{\underset{K,k}{\Longrightarrow}} A'\sigma$ *(resp.* $A\sigma \Longrightarrow A'\sigma$*).*

3. $A\sigma \xrightarrow[K,k]{\mu} A'$ *implies* $A \xrightarrow[K,k]{\mu'} A''$*, with $\mu'\sigma = \mu$ and $A''\sigma = A'$.*

4. $A\sigma \xrightarrow{\tau} A'$ *implies either:*

   (a) there exists A_1 s.t. $A \xrightarrow{\tau} A_1$ *with $A_1\sigma = A'$, or*

   (b) there exist two transitions $A \xrightarrow[K_1,k_1]{a(\widetilde{b}_0)} A_1 \xrightarrow[K_2,k_2]{\boldsymbol{\nu}\,\widetilde{b}'\,\overline{c}\langle\widetilde{b}\rangle} A_2$ *with k_1 and k_2 fresh for A and A_1, respectively, $k_1 \notin K_2$, $a\sigma = c\sigma$ and* $A' \equiv \boldsymbol{\nu}\,\widetilde{b}'\,(A_2\rho\sigma\{\widetilde{b}\sigma/\widetilde{b}_0\})$*, where* $\rho \stackrel{\text{def}}{=} [k_1 \rightsquigarrow K_2, k_2 \rightsquigarrow K_1]$.

PROOF: Items 1, 3 and 4 are proven by straightforward transition induction. Item 2 is a consequence of item 1. $\square$

Our next task is to prove a kind of "converse" of part (4.b) of the previous lemma for weak transitions, whereby two independent transitions, possibly interleaved with some silent transitions, are composed. This will be done in Lemma 4.8. In its proof, we shall use Lemma 4.7, asserting that, given a sequence of silent transitions from $K :: B \mid A$, process A can be decomposed into two subprocesses, one actually interacting with B (the process called A_1 below), the other evolving on its own (the process called A_2).

Lemma 4.7 *Let A and B be causal processes and suppose that* $K :: B \mid A \Longrightarrow C$*. Then for some $\widetilde{e}$, A_1, A_2, $\widetilde{d}$, B', A_1' and A_2' we have:*

a) $A \equiv \boldsymbol{\nu}\,\widetilde{e}\,(A_1 \mid A_2)$;

b) $K :: B \mid A_1 \Longrightarrow\equiv (\boldsymbol{\nu}\,\widetilde{d})K :: (B' \mid A_1')$;

c) $A_2 \Longrightarrow\equiv A_2'$;

d) $C \equiv (\boldsymbol{\nu}\,\widetilde{d}, \widetilde{e})\big(K :: (B' \mid A_1') \mid A_2'\big)$.

PROOF: It must be that $K :: B \mid A \xrightarrow{\tau^m} C$, for some $m \geq 0$. The proof goes by induction on m. The case $m = 0$ is trivial, thus suppose $m > 0$. Therefore we have $K :: B \mid A \xrightarrow{\tau} E \xrightarrow{\tau^{m-1}} C$, for some E. We analise the first τ-step, then the remaining $m-1$ steps, and finally we explain that the concatenation of these steps give the required decomposition.

We distinguish the possible ways in which the transition $K :: B \mid A \xrightarrow{\tau} E$ may arise. The cases when it arises from $K :: B$ alone or from A alone are easily dealt with by exploiting the induction hypothesis. We treat in detail only the case when the transition arises as an interaction between $K :: B$ and A. Applying the Structural Lemma 4.5 to the interacting transitions, we can individuate the subcomponents of $K :: B$ and A, respectively F_1 and F_2, from which these transitions arise. Formally, it must be:

$$\begin{aligned} K :: B &\equiv K :: \boldsymbol{\nu}\, \widetilde{d}_1\, (F_1 \mid D_1) && (1)\\ A &\equiv \boldsymbol{\nu}\, \widetilde{d}_2\, (F_2 \mid D_2) && (2)\\ E &\equiv (\boldsymbol{\nu}\, \widetilde{d}_1, \widetilde{d}_2)\big(K :: (F_1' \mid F_2' \mid D_1) \mid D_2\big)\,. && (3)\end{aligned}$$

where

$$K :: F_1 \mid F_2 \xrightarrow{\tau} \equiv K :: (F_1' \mid F_2') \qquad (4)$$

Define now $B_* \stackrel{\text{def}}{=} K :: (F_1' \mid F_2' \mid D_1)$ and $A_* \stackrel{\text{def}}{=} D_2$. Since $E \xrightarrow{\tau^{m-1}} C$, from (3) we deduce that it must be $C \equiv (\boldsymbol{\nu}\, \widetilde{d}_1, \widetilde{d}_2)C_*$, where

$$K :: B_* \mid A_* \xrightarrow{\tau^{m-1}} C_*\,. \qquad (5)$$

Applying the induction hypothesis to (5), we obtain:

$$\begin{aligned} A_* &\equiv \boldsymbol{\nu}\, \widetilde{e_*}\, (A_{1*} \mid A_{2*}), && (6)\\ (K :: B_*) \mid A_{1*} &\Longrightarrow\equiv \boldsymbol{\nu}\, \widetilde{d}_*\, K :: (B_*' \mid A_{1*}'), && (7)\\ A_{2*} &\Longrightarrow\equiv A_{2*}', && (8)\\ C_* &\equiv (\boldsymbol{\nu}\, \widetilde{d}_*, \widetilde{e_*})\big(K :: (B_*' \mid A_{1*}') \mid A_{2*}'\big). && (9)\end{aligned}$$

We now define the following expressions:

$$\begin{array}{lll} A_1 \stackrel{\text{def}}{=} F_2 \mid A_{1*} & A_1' \stackrel{\text{def}}{=} A_{1*}' & \widetilde{e} \stackrel{\text{def}}{=} \widetilde{d}_2\widetilde{e_*} \\ A_2 \stackrel{\text{def}}{=} A_{2*} & A_2' \stackrel{\text{def}}{=} A_{2*}' & \widetilde{d} \stackrel{\text{def}}{=} \widetilde{d}_*\widetilde{d}_1 \\ B' \stackrel{\text{def}}{=} B_*'\,. & & \end{array}$$

With these definitions, it is simple to prove assertions (a), (b), (c) and (d) of the lemma. As an example, we verify (b).

$$\begin{array}{lll} K :: B \mid A_1 & \equiv \boldsymbol{\nu}\, \widetilde{d}_1\, (K :: F_1 \mid F_2 \mid K :: D_1 \mid A_{1*}) & \text{((1), definition of } A_1 \text{ and rules for } \equiv) \\ & \xrightarrow{\tau}\equiv \boldsymbol{\nu}\, \widetilde{d}_1\, (K :: (F_1' \mid F_2' \mid D_1) \mid A_{1*}) & \text{((4) and rules for } \equiv) \\ & \equiv \boldsymbol{\nu}\, \widetilde{d}_1\, (K :: B_* \mid A_{1*}) & \text{(definition of } B_*) \\ & \Longrightarrow\equiv (\boldsymbol{\nu}\, \widetilde{d}_1, \widetilde{d}_*)K :: (B_*' \mid A_{1*}') & \text{(assertion (7))} \\ & \equiv \boldsymbol{\nu}\, \widetilde{d}\, K :: (B' \mid A_1') & \text{(definitions of } B',\ A_1' \text{ and } \widetilde{d}). \end{array}$$

□

We are now ready to prove the "converse" of item 4 of Lemma 4.6.

Lemma 4.8 *Let B be a causal process, and suppose that $B \overset{a(\widetilde{b}_0)}{\underset{K_1,k_1}{\Longrightarrow}} B'' \overset{\boldsymbol{\nu}\,\widetilde{b}'\,\overline{a}\langle\widetilde{b}\rangle}{\underset{K_2,k_2}{\Longrightarrow}} B'$, with $\widetilde{b}_0$, $\widetilde{b}'$, k_1 fresh for B, k_2 fresh for B'' and $k_1 \notin K_2$. Then $B \Longrightarrow\equiv \boldsymbol{\nu}\, \widetilde{b}'\, (B'\rho\sigma)$, with $\sigma = \{\widetilde{b}/\widetilde{b}_0\}$ and $\rho = [k_1 \rightsquigarrow K_2,\ k_2 \rightsquigarrow K_1]$.*

PROOF: Transitions $B \overset{a(\widetilde{b}_0)}{\underset{K_1,k_1}{\Longrightarrow}} B'' \overset{\boldsymbol{\nu}\,\widetilde{b}'\,\overline{a}\langle\widetilde{b}\rangle}{\underset{K_2,k_2}{\Longrightarrow}} B'$ can be decomposed thus:

$$B \Longrightarrow B_1 \xrightarrow[K_1,k_1]{a(\widetilde{b}_0)} B_2 \Longrightarrow B_3 \xrightarrow[K_2,k_2]{\boldsymbol{\nu}\,\widetilde{b}'\,\overline{a}\langle\widetilde{b}\rangle} B_4 \Longrightarrow B'\,.$$

Applying the Structural Lemma 4.5 to transition $B_1 \xrightarrow[K_1,k_1]{a(\widetilde{b}_0)} B_2$, we infer that there are $S, P, A, \widetilde{d}$ such that:

$$B_1 \equiv \boldsymbol{\nu}\,\widetilde{d}\big(K_1 :: (S + a(\widetilde{b}_0).P) \mid A\big) \tag{10}$$
$$B_2 \equiv \boldsymbol{\nu}\,\widetilde{d}\big((K_1,k_1) :: P \mid A\big)\,. \tag{11}$$

Applying Lemma 4.7 to transition $B_2 \Longrightarrow B_3$, due to the form of B_2, we infer that there are $\widetilde{e}, A_1, A_2, \widetilde{d}', P', A_1'$ such that:

$$A \equiv \boldsymbol{\nu}\,\widetilde{e}\,(A_1 \mid A_2) \tag{12}$$
$$(K,k_1) :: P \mid A_1 \Longrightarrow\equiv \boldsymbol{\nu}\,\widetilde{d}'\,(K,k_1) :: (P' \mid A_1') \tag{13}$$
$$A_2 \Longrightarrow\equiv A_2' \tag{14}$$
$$B_3 \equiv (\boldsymbol{\nu}\,\widetilde{d},\widetilde{d}',\widetilde{e})\big((K_1,k_1) :: (P' \mid A_1') \mid A_2'\big)\,. \tag{15}$$

Let us consider now transition $B_3 \xrightarrow[K_2,k_2]{\boldsymbol{\nu}\,\widetilde{b}'\,\overline{a}\langle\widetilde{b}\rangle} B_4$: since $k_1 \notin K_2$, from (15) we deduce that this transition originates from A_2'. Formally, we have:

$$B_4 \equiv \boldsymbol{\nu}\,\widetilde{f}_1\,\big((K_1,k_1) :: (P' \mid A_1') \mid A_2''\big) \text{ where } \{\widetilde{f}_1\} = \{\widetilde{d},\widetilde{d}',\widetilde{e}\} - \{\widetilde{b}'\}, \text{ and} \tag{16}$$
$$A_2' \xrightarrow[K_2,k_2]{\boldsymbol{\nu}\,\widetilde{f}_2\,\overline{a}\langle\widetilde{b}\rangle} A_2'' \qquad \text{where } \{\widetilde{f}_2\} = \{\widetilde{b}'\} - \{\widetilde{d},\widetilde{d}',\widetilde{e}\}\,. \tag{17}$$

We now prove that $B_1 \Longrightarrow\equiv \boldsymbol{\nu}\,\widetilde{b}'\,(B'\rho\sigma)$, as follows:

$$\begin{array}{rll}
B_1 & \equiv (\boldsymbol{\nu}\,\widetilde{d},\widetilde{e})\big(K_1 :: (S + a(\widetilde{b}_0).P) \mid A_1 \mid A_2\big) & \text{(by (10) and (12))} \\
& \Longrightarrow\equiv (\boldsymbol{\nu}\,\widetilde{d},\widetilde{e})\big(K_1 :: (S + a(\widetilde{b}_0).P) \mid A_2' \mid A_1\big) & \text{(by (14))} \\
& \xrightarrow{\tau}\equiv (\boldsymbol{\nu}\,\widetilde{d},\widetilde{e},\widetilde{f}_2)\big((K_1,K_2) :: P\{\widetilde{b}/\widetilde{b}_0\} \mid A_2''[k_2 \rightsquigarrow K_1] \mid A_1\big) & \text{(by (17) and rule \texttt{com})} \\
& \equiv (\boldsymbol{\nu}\,\widetilde{d},\widetilde{e},\widetilde{f}_2)\Big(\big((K_1,k_1) :: P \mid A_1 \mid A_2''\big)\sigma\Big)\rho & \text{(by def. of } \sigma \text{ and } \rho) \\
& \Longrightarrow\equiv (\boldsymbol{\nu}\,\widetilde{d},\widetilde{d}',\widetilde{e},\widetilde{f}_2)\Big(\big((K_1,k_1) :: (P' \mid A_1') \mid A_2''\big)\sigma\Big)\rho & \text{(by (13) and lemmas 4.6.2 and 4.4)} \\
& \equiv (\boldsymbol{\nu}\,\widetilde{b}',\widetilde{f}_1)\Big(\big((K_1,k_1) :: (P' \mid A_1') \mid A_2''\big)\sigma\Big)\rho & \text{(since } \{\widetilde{d},\widetilde{d}',\widetilde{e},\widetilde{f}_2\} = \{\widetilde{b}',\widetilde{f}_1\}) \\
& \equiv \boldsymbol{\nu}\,\widetilde{b}'\,\Big(\big(\boldsymbol{\nu}\,\widetilde{f}_1\,((K_1,k_1) :: (P' \mid A_1') \mid A_2'')\big)\sigma\Big)\rho & \text{(definitions of } \widetilde{f}_1 \text{ and } \sigma) \\
& \equiv \boldsymbol{\nu}\,\widetilde{b}'\,(B_4\sigma)\rho & \text{(by (16))} \\
& \Longrightarrow\equiv \boldsymbol{\nu}\,\widetilde{b}'\,(B'\sigma)\rho & \text{(by } B_4 \Longrightarrow B' \text{ and lemmas 4.6.2} \\
& \equiv \boldsymbol{\nu}\,\widetilde{b}'\,(B'\rho\sigma) & \quad \text{and 4.4).}
\end{array}$$

Putting together $B \Longrightarrow B_1$ and $B_1 \Longrightarrow\equiv \boldsymbol{\nu}\,\widetilde{b}'\,(B'\rho\sigma)$ we get the thesis. □

A simple proof technique for bisimilarity:

Definition 4.9 *A symmetric binary relation $\mathcal{R}$ over causal processes is a* ground causal bisimulation up to $\equiv$ and restriction *if, whenever $A\,\mathcal{R}\,B$ and $A \xrightarrow[K,k]{\mu} A'$ (resp. $A \xrightarrow{\tau} A'$) with k fresh for A and B, there exist $\widetilde{b}$, A_1, B' and B_1 s.t.:*

$$B \overset{\mu}{\underset{K,k}{\Longrightarrow}} B' \textit{ (resp. } B \Longrightarrow B'\textit{) and } A' \equiv \boldsymbol{\nu}\,\widetilde{b}\,A_1 \textit{ and } B' \equiv \boldsymbol{\nu}\,\widetilde{b}\,B_1 \textit{ and } A_1\,\mathcal{R}\,B_1\,.$$

Lemma 4.10 *If $\mathcal{R}$ is a ground causal bisimulation up to $\equiv$ and restriction then $\mathcal{R} \subseteq \approx^{\mathrm{c}}_{\mathrm{g}}$.*

4.2 The congruence results

Theorem 4.11 **($\approx^{c}_{g}$ is preserved by substitutions)** *Let A, B be causal processes and σ be a substitution. Then $A \approx^{c}_{g} B$ implies $A\sigma \approx^{c}_{g} B\sigma$.*

PROOF: We show that

$$\mathcal{R} = \{(A\sigma, B\sigma) \mid \sigma \text{ is a substitution and } A \approx^{c}_{g} B\}$$

is a ground causal bisimulation up to $\equiv$ and restriction. We have to check that whenever $A\sigma \,\mathcal{R}\, B\sigma$ and $A\sigma \xrightarrow[K,k]{\mu} A'$ (resp. $A\sigma \xrightarrow{\tau} A'$), with k fresh, then we can find A'', σ', B'' and $\widetilde{b}'$ s.t.:

$$B\sigma \underset{K,k}{\overset{\mu}{\Longrightarrow}} B' \text{ (resp. } B\sigma \Longrightarrow B') \text{ and } B' \equiv \boldsymbol{\nu}\,\widetilde{b}'\,(B''\sigma') \text{ and } A' \equiv \boldsymbol{\nu}\,\widetilde{b}'\,(A''\sigma') \text{ and } \text{ and } A'' \approx^{c}_{g} B''\,. \tag{18}$$

We only deal with the case when $A\sigma \xrightarrow{\tau} A'$. The case $A\sigma \xrightarrow[K,k]{\mu} A'$ can be dealt with using Lemma 4.6(1-3). Thus, suppose $A\sigma \xrightarrow{\tau} A'$. According to Lemma 4.6.4, the two sub-cases (a) and (b) may arise. We consider the latter, which is more difficult. Therefore we have $A \xrightarrow[K_1,k_1]{a'(\widetilde{b}_0)} A_1 \xrightarrow[K_2,k_2]{\boldsymbol{\nu}\,\widetilde{b}'\,\overline{a''}\langle\widetilde{b}\rangle} A_2$, with $a'\sigma = a''\sigma = a$ and $A' \equiv \boldsymbol{\nu}\,\widetilde{b}'\,(A_2\rho\sigma')$ and $\rho \stackrel{\text{def}}{=} [k_1 \rightsquigarrow K_2, k_2 \rightsquigarrow K_1]$ and $\sigma' \stackrel{\text{def}}{=} \sigma\{\widetilde{b}\sigma/\widetilde{b}_0\}$. Since $A \approx^{c}_{g} B$, there are B_1 and B_2 s.t. the diagram below commutes:

$$\begin{array}{ccccc}
A & \xrightarrow[K_1,k_1]{a'(\widetilde{b}_0)} & A_1 & \xrightarrow[K_2,k_2]{\boldsymbol{\nu}\,\widetilde{b}'\,\overline{a''}\langle\widetilde{b}\rangle} & A_2 \\
\approx^{c}_{g} & & \approx^{c}_{g} & & \approx^{c}_{g} \\
B & \underset{K_1,k_1}{\overset{a'(\widetilde{b}_0)}{\Longrightarrow}} & B_1 & \underset{K_2,k_2}{\overset{\boldsymbol{\nu}\,\widetilde{b}'\,\overline{a''}\langle\widetilde{b}\rangle}{\Longrightarrow}} & B_2\,.
\end{array}$$

From these transitions of B and Lemma 4.6(2) we infer

$$B\sigma \underset{K_1,k_1}{\overset{a(\widetilde{b}_0)}{\Longrightarrow}} B_1\sigma \underset{K_2,k_2}{\overset{\boldsymbol{\nu}\,\widetilde{b}'\,\overline{a}\langle\widetilde{b}\sigma\rangle}{\Longrightarrow}} B_2\sigma.$$

Since $k_1 \notin K_2$, we can apply Lemma 4.8 and infer:

$$B\sigma \Longrightarrow B' \equiv \boldsymbol{\nu}\,\widetilde{b}'\,(B_2\rho\sigma\{\widetilde{b}\sigma/\widetilde{b}_0\}) \equiv \boldsymbol{\nu}\,\widetilde{b}'\,(B_2\rho\sigma')\,.$$

We prove that (A', B') belongs to $\mathcal{R}$, up to the restriction $\boldsymbol{\nu}\,\widetilde{b}'$, by exhibiting A'' and B'' such that (18) is fulfilled. Now, from $A_2 \approx^{c}_{g} B_2$ and Lemma 4.4.2, it follows that $A'' \stackrel{\text{def}}{=} A_2\rho \approx^{c}_{g} B_2\rho \stackrel{\text{def}}{=} B''$. Therefore, by definition of $\mathcal{R}$, we have $A''\sigma' \,\mathcal{R}\, B''\sigma'$. But $A' \equiv \boldsymbol{\nu}\,\widetilde{b}'\,(A''\sigma')$ and $B' \equiv \boldsymbol{\nu}\,\widetilde{b}'\,(B''\sigma')$, thus (18) holds. □

As easy corollaries of the above theorem, we get:

Corollary 4.12 *$\approx^{c}_{g}$ is a congruence relation in the language $\mathcal{P}$.*

PROOF: One shows that $\approx^{c}_{g}$ is preserved by each operator of the language $\mathcal{P}$, by exhibiting appropriate bisimulations. For input prefix and parallel composition, one exploits the fact that $\approx^{c}_{g}$ is preserved by name substitutions. □

Corollary 4.13 *Ground, late, early and open forms of weak causal bisimilarity coincide in the language $\mathcal{P}$.*

PROOF: As an example, we consider the proof that $\approx^{c}_{g}$ and $\approx^{c}_{e}$ coincide. The inclusion $\approx^{c}_{e} \subseteq \approx^{c}_{g}$ follows by definition (the requirements in the definition of $\approx^{c}_{g}$ are also in the definition of $\approx^{c}_{e}$). For the converse, one shows that $\approx^{c}_{g}$ is a causal bisimulation; to satisfy the input clause of Definition 4.1 one uses the fact that $\approx^{c}_{g}$ is closed under substitutions. □

For *strong* causal bisimulation, the same results of the weak case hold. The proof schema is similar but the proofs are simpler (for instance, Lemma 4.8 is not needed).

Corollary 4.14

1. *Strong ground causal bisimilarity is a congruence relation in the language $\mathcal{P}$;*

2. *ground, late, early and open forms of strong causal bisimilarity coincide in the language $\mathcal{P}$.*

The forms of causal bisimilarities considered in this paper explicitly reveal the causal dependencies induced by the nesting of prefixes and propagated through communication, and called *subject dependencies* in [2]. There exist another form of causal dependency in the π-calculus, induced by the binding mechanism on names. As an example, in $\boldsymbol{\nu}\, b\, (\overline{a}\langle b\rangle \,|\, b(x))$ the execution of the output at a opens the scope of the restriction $\boldsymbol{\nu}\, b$, thus enabling the execution of $b(x)$, which was previously blocked. In [2], this form of causality is called *object* causality. By contrast with subject dependencies, object dependencies are directly revealed in the standard interleaving transition systems. Various ways of combining subject and object dependencies are possible, and lead to different causal relations on processes (see [11] for a survey). Our choice of handling subject dependencies in isolation is due to two main reasons: First, the definitions of the bisimulations are simpler. Secondly, subject dependencies are essential for the congruence results studied in this paper. We think that the same results hold for other causal equivalences which take subject dependencies into account.

Acknowledgements

The detailed comments of the two anonymous referees allowed us to improve the technical presentation and correct a number of typos.

References

[1] N. G. de Bruijn. Lambda-calculus notation with nameless dummies: a tool for automatic formula manipulation with application to the Church-Rosser theorem. *Indag. Math.*, 34(5):381–392, 1972.

[2] M. Boreale and D. Sangiorgi. A fully abstract semantics for causality in the π-calculus. Technical Report ECS–LFCS–94–297, LFCS, Dept. of Comp. Sci., Edinburgh Univ., 1994. To appear in *Acta Informatica*. An extract appeared in *Proc. STACS'95*, LNCS 900, Springer Verlag.

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

[4] M. Hansen, H. Hüttel, and J. Kleist. Bisimulations for asynchronous mobile processes. In *Proc. Tbilisi Symposium on Language, Logic, and Computation*, 1996. Also available as BRICS Report No. EP-95-HHK, Aalborg University, Denmark 1996.

[5] K. Honda. Two bisimilarities for the ν-calculus. Technical Report 92-002, Keio University, 1992.

[6] K. Honda and M. Tokoro. On asynchronous communication semantics. In M. Tokoro, O. Nierstrasz, P. Wegner, and A. Yonezawa, editors, *ECOOP '91 Workshop on Object Based Concurrent Programming,* Geneva, Switzerland, 1991 , volume 612 of *Lecture Notes in Computer Science*, pages 21–51. Springer Verlag, 1992.

[7] A. Kiehn. Comparing locality and causality based equivalences. *Acta Informatica*, 31:697–718, 1994. Revision of *Local and global causes*, report TUM–I9132, 1991.

[8] 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.

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

[10] B. C. Pierce, D. Rémy, and D. N. Turner. A typed higher-order programming language based on the pi-calculus. In *Workshop on Type Theory and its Application to Computer Systems, Kyoto University*, 1993.

[11] C. Priami. *Enhanced Operational Semantics for Concurrency.* PhD thesis, Department of Computer Science, Università di Pisa, 1995.

[12] D. Sangiorgi. A theory of bisimulation for the π-calculus. *Acta Informatica*, 33:69–97, 1996. Extended Abstract in *Proc. CONCUR '93*, volume 715 of Lecture Notes in Computer Science, Springer Verlag.

[13] D. Sangiorgi. On the bisimulation proof method. Technical Report ECS–LFCS–94–299, LFCS, Dept. of Comp. Sci., Edinburgh Univ., 1994.

[14] D. Sangiorgi. Lazy functions and mobile processes. Technical Report RR-2515, INRIA-Sophia Antipolis, 1995. To appear in the "Festschrift volume in honor of Robin Milner's 60th birthday", Cambridge Press.

[15] D. Sangiorgi. πI: A symmetric calculus based on internal mobility. In Mosses P. et al. editor, *Proc. TAPSOFT'95*, volume 915 of *Lecture Notes in Computer Science*, pages 172–186. Springer Verlag, 1995. Full version in *Theoretical Computer Science* 167(2):235–274, 1996.