A fully abstract semantics for causality in the π-calculus

Michele Boreale
Dipartimento di Scienze dell'Informazione
Università di Roma "La Sapienza"

Davide Sangiorgi
Department of Computer Science
University of Edinburgh

Abstract

We examine the meaning of causality in calculi for mobile processes like the π-calculus, and we investigate the relationship between interleaving and causal semantics for such calculi.

We separate two forms of causal dependencies on actions of π-calculus processes, called *subject* and *object* dependencies: The former originate from the nesting of prefixes and are propagated through interactions among processes (they are the only form of causal dependencies present in CCS-like languages); the latter originate from the binding mechanisms on names. We propose a notion of causal bisimulation which distinguishes processes which differ for the subject or for the object dependencies. We show that this *causal* equivalence can be reconducted to, or implemented into, the ordinary *interleaving* observation equivalence. We prove that our encoding is fully abstract w.r.t. the two behavioural equivalences. This allows us to exploit the simpler theory of the interleaving semantics to reason about the causal one.

In [San94b] a similar programme is carried out for *location bisimulation* [BCHK91], a non-interleaving spatial-sensitive (as opposed to causal-sensitive) behavioural equivalence. The comparison between the encodings of causal bisimulation in this paper, and of location bisimulation in [San94b], evidences the similarities and the differences between these two equivalences.

1 Introduction

In the field of semantics for process algebras, a major division is between *interleaving* semantics and *non-interleaving* semantics. In the former case, only the temporal dependencies of action executions are observed; in the latter case also either the causal dependencies among actions (*causal* semantics) or the spatial distribution of systems (*location* semantics) are considered. Non-interleaving semantics give a better account of the concurrency and of the dependencies among the activities of a system. But the easier mathematics of interleaving semantics has permitted the development, for instance, of simpler algebraic characterisations and modal logics.

Lately, there has been a growth of interest for *name-passing* process algebras, in which the possibility of communicating *names* (a synonymous for 'channels') permits the modeling of systems with dynamic linkage reconfigurations. The best known of these calculi is the *π-calculus* [MPW92]. Most theoretical research on the π-calculus has dealt with interleaving semantics [MPW92, BD92, PS93, San92, Wal94]; a major concern has been investigation of its expressiveness. The non-interleaving side remains largely unexplored; only very recently, the analysis of a location semantics has appeared [San94b].

The main goal of this paper is to investigate the the relationship between interleaving and causal semantics for the π-calculus. We focus on the interleaving *observation equivalence* (also called *weak bisimulation*), written $\approx$ [Mil89, MPW92], and on the non-interleaving *causal bisimulation*, written $\approx_{\mathrm{c}}$ [DDM88, DD89, Kie91]; these are among the most studied behavioural equivalences in the respective groups. W.r.t. observation equivalence, causal bisimulation makes additional distinctions on processes on the basis of the causal dependencies for their actions. Our major result shows that causal bisimulation can be reconducted to, or implemented into, observation equivalence. This allows us to use the simpler theory of the interleaving semantics to reason about the causal one; the result also evidences the expressive power of the π-calculus.

Causal bisimulation represents quite a stable relation. Several approaches to causality have independently led to it. Examples include Degano, De Nicola and Montanari' *history preserving bisimulation* [DDM88], Darondeau and Degano' *causal-tree bisimulation* [DD89] and Kiehn's *global-cause bisimulation* [Kie91]. (Comparisons of causal bisimulation with other interleaving and non-interleaving equivalences can be found in [Ace92b, Ace92a, DDM93].) Known descriptions of causal bisimulation over CCS-like languages are based on operational transition rules more complex than the standard interleaving ones [Mil89], because they carry information about causal dependencies, in form of pointer-like objects among actions. This extra structure also has to be lifted at the syntactic level, forcing an extension of the basic language; existing algebraic theories and proof systems of causal bisimulation [DD89, Kie93] heavily rely on the added causal operators. A motivation for trying to implement causal bisimulation at the interleaving level is to seek ways for avoiding the introduction of this extra structure.

Another goal of this paper is to investigate the meaning of causality in calculi for mobile processes. We have individuated two forms of dependencies between actions, which we have called *subject* and *object* dependencies. Subject dependencies originate from the nesting of prefixes and are propagated through internal communications of processes. For example, the processes $a.b.\mathbf{0} + b.a.\mathbf{0}$ can perform actions a and b in either order. In either case, there is a dependency between the two actions — the firing of the first enables the firing of the second. The propagation of subject dependencies through interactions is shown in $\boldsymbol{\nu}\, c\,(a.c.\mathbf{0} \mid \overline{c}.b.\mathbf{0})$ (we use the π-calculus notation $\boldsymbol{\nu}\, a\, P$ for the restriction of name a in process P). In this process, the interaction at c creates a dependency of b from a; indeed, it holds that $\boldsymbol{\nu}\, c\,(a.c.\mathbf{0} \mid \overline{c}.b.\mathbf{0})$ is causally equivalent with $a.b.\mathbf{0}$. In CCS-like languages, i.e., languages in which actions represent pure synchronisation, subject dependencies are the only form of causal dependencies. Subject dependencies cannot be detected at the interleaving level. For example, one cannot detect the dependency between the actions a and b of the process $a.b.\mathbf{0} + b.a.\mathbf{0}$. Hence, in an interleaving semantics this process is equated to the process $a.\mathbf{0} \mid b.\mathbf{0}$; this equality is rejected in a causal semantics because in $a.\mathbf{0} \mid b.\mathbf{0}$ actions a and b are independent.

The other form of dependencies, namely the object dependencies, are determined by π-calculus binding mechanisms on actions; therefore they do not exist in CCS. Approximately, an action λ is object dependent from a previous action μ if μ acts as binder for some name of λ; as such, μ can be thought of as supplying some names to λ. There are two binding actions in the π-calculus, namely input and bound output — the latter denoting the output of a restricted, i.e., private, name.

By contrast with subject dependencies, the possible object dependencies in a process can be detected at the interleaving level, simply revealed by the occurrences of names in actions. For instance, consider the bound output transition $P_1 \xrightarrow{(\boldsymbol{\nu}\, b)\overline{a}\langle b\rangle} P_2$, which says that P_1 can perform the output at a of the private name b and become P_2 in doing so; the object dependencies from the action $(\boldsymbol{\nu}\, b)\overline{a}\langle b\rangle$ are given by those actions in the successive behaviour of P_2 in which b occurs free. A similar consideration can be made for input transitions $P_1 \xrightarrow{a(b)} P_2$ — which says that P_1 can perform the input at a of b so becoming P_2 — if name b is not already present in P_1. In other words, observationally equivalent processes show the same object dependencies.

In most of the cases, an object dependency is also a subject dependency (for instance, this is always the case for the object dependencies arising from input actions). There are, however, exceptions. For instance, the process $P \overset{def}{=} \boldsymbol{\nu}\, b\left(\overline{a}\langle b\rangle.\, \mathbf{0} \mid \overline{b}\langle y\rangle.\, \mathbf{0}\right)$ can perform the output at a of the restricted name b and then an output at b; the former action enables the latter, because it "opens" the restriction at b. There is an object dependency of the action at b from the action at a, but no subject dependency.

In this paper, we have chosen to separate subject dependencies from object dependencies, because they represent logically different forms of causality. Hence we shall want to distinguish processes which differ for the subject or for the object dependencies. For instance, we shall discriminate between the process P above and the process $Q \overset{def}{=} \boldsymbol{\nu}\, b\,(\overline{a}\langle b\rangle.\, \overline{b}\langle y\rangle.\, \mathbf{0})$. Both processes can perform an action at a and then an action at b. However, in Q there is a double causal dependency between the two actions: One given by a prefix-nesting (i.e., a subject dependency), and the other given by a name-binding (i.e., an object dependency). By contrast, in P there is a single dependency between the two actions, given by name-binding.

To define a causal bisimulation which reflects these discriminations on processes, we enrich the standard (i.e., interleaving) transition system of the π-calculus with information about the subject dependencies. This is enough because the object dependencies can already be detected in the standard system. Our enriched system is defined following Kiehn's system [Kie91], which describes causal bisimulation in CCS (as noted above, in CCS all causal dependencies are subject dependencies). This implies adding explicit causes to π-calculus syntax and operational semantics. Causal bisimulation is defined on top of the enriched system using familiar bisimulation techniques.

We then show that the sophisticated observational machinery of causal bisimulation ($\approx_{\mathrm{c}}$) can be represented from within the interleaving observation equivalence ($\approx$). We define a compositional encoding $[\![\,.\,]\!]$ from the enriched calculus to the basic one, and we prove that it is fully abstract w.r.t. $\approx_{\mathrm{c}}$ and $\approx$; i.e., for each pair of terms A and B in the enriched calculus we have:

$$A \approx_{\mathrm{c}} B \text{ if and only if } [\![A]\!] \approx [\![B]\!] .$$

Through various examples, we show how this result and the algebraic theory of $\approx$ can be used to prove properties about $\approx_{\mathrm{c}}$. When restricted to the sublanguage with no object dependencies, that is CCS, our encoding yields a characterisation of the classic causal bisimulation of [DD89, Kie91] in terms of the observation equivalence of the monadic π-calculus.

In the definition of encoding, we exploit an important feature of the π-calculus: The calculus naturally permits the description of *data structures* which can be created and combined at run time. These data structures are modeled as standard processes accessible via private names, which can be passed around and used to form more complex structures. We use data structures called *wires* to encode the observability of the explicit causes used in the definition of causal bisimulation. A wire receives names at an *entrance-point* and retransmit them at a bunch of *end-points*. Several wires can be connected together, connection representing union of causes. Reachability among wires, i.e. if an end-point of a wire can be reached from the entrance-point of another wire, encodes the subject dependencies between actions. When a target process $[\![A]\!]$ performs a visible action μ, a wire is generated and a pointer to the wire — under the form of a fresh name — is emitted. This pointer can be used to 'explore' the wire and determine the subject dependencies for μ. Transmission of causes across a parallel composition, a distinguishing feature of causal bisimulation, is rendered as an exchange of a private name, by which two previously separated wires get connected. The proof of our full abstraction theorem exploits a few properties of concatenation of wires that permit symbolic manipulations of these structures.

The paper mainly relates to [San94b]. There, a programme similar to that of the present paper is carried out for *location bisimulation* [BCHK91], one of the most convincing spatial-sensitive semantics. An encoding from an enriched π-calculus of *located processes* to the standard π-calculus is given and proved to be fully abstract w.r.t. location bisimulation and observation equivalence. The treatment of causes in communications is what distinguishes causal bisimulation from location bisimulation: In the former, the causes of the interacting actions must be appropriately merged, whereas in the latter they remain separate. This explains the difference between the data structures (the wires) used in the encoding of the two equivalences, here and in [San94b]. In the case of causal bisimulation, the connections among wires, established as the computation proceeds, may create *branching* chains; by contrast, in the case of location bisimulation connections among wires may only create *linear* chains. The difference also shows up in the proofs: Although the overall structure of the proofs of full abstraction is similar in the two papers, the technical details are quite different, sometimes strikingly so. A thorough comparison between causal and location bisimulation via their encodings into observation equivalence presented here and in [San94b] is in Section 10. The outcomes of this comparison are certainly not new. Careful studies of the relationship between causal- and spatial-sensitive equivalences can be found in the literature [Ace92b, CD93, MY92, Kie91, Kie93, DP92]. The use of the encodings gives us a refreshing perspective on the issue.

The remainder of the paper is organised as follows. In Section 2, we review the syntax of the standard polyadic π-calculus and the definition of observation equivalence. In Section 3 we present the proof techniques (mainly various forms of "bisimulation up-to") that will be extensively used in later sections. In Section 4 we introduce the enriched language of causal terms, and we define causal bisimulation. In Section 5 we present the encoding $[\![\,.\,]\!]$. Its full abstraction is proved in Sections 6 through 8, which are organised as follows. In Section 6, some important properties of wire processes are shown. These properties are used in Section 7, to establish the operational correspondence between transitions of causal terms and transitions of encoded processes, and in Section 8, to derive the main theorems. In Section 9 we show some examples of applications of the encoding. In Section 10 causal bisimulation and location bisimulation are compared using their encodings into observation equivalence presented here and in [San94b]. The concluding section mentions directions for possible future research.

2 The polyadic π-calculus

We start by reviewing the standard syntax and the interleaving semantics of the polyadic π-calculus, as in [Mil91].

2.1 Syntax of standard processes

Definition 2.1 (Standard processes) *Consider an infinite countable set* $\mathcal{N}$ *of names s.t.* $\mathcal{N} \cap \{\tau\} = \emptyset$ *. The class of the* polyadic standard processes over names $\mathcal{N}$, *written*

$\mathcal{P}(\mathcal{N})$, is built from the operators of inaction, input prefix, output prefix, silent prefix, sum, parallel composition, restriction, and replication:

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

In the sequel, $a, b, \ldots, x, y, \ldots$ range over names, and P, Q, R over standard processes. 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 $a(\tilde{b}).\,P$, where $\tilde{b}$ has pairwise distinct components, 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 simultaneous 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. A silent prefix $\tau.P$ denotes a process which may evolve to P without interactions with the environment. 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. A replication $!\,P$ stands for a countable infinite number of copies of P in parallel. We write $\Pi_{i \in I} P_i$ as an abbreviation for $P_{i_1} \mid \ldots \mid P_{i_n}$, $I = \{i_1, \ldots, i_n\}$. Symbol I will range over finite indexing sets. We assign parallel composition and sum the lowest precedence among the operators.

In prefixes $a(\tilde{b})$ and $\overline{a}\langle\tilde{b}\rangle$, we call a the *subject* and $\tilde{b}$ the *object.* We use α to range over prefixes and often abbreviate $\alpha.\mathbf{0}$ as α. The operators $a(\tilde{b}).P$ and $\boldsymbol{\nu}\, b\, P$ bind all free occurrences of names $\tilde{b}$ and b in P. These binders give rise in the expected way to the definition of *free names* of a term. The definitions of *name 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". Sometimes, we use $\stackrel{def}{=}$ as abbreviation mechanism, to assign a name to an expression to which we want to refer later. In an assertion, we say that a name is *fresh* to mean that it it is different from any other name which is nominated in the assertion or which occurs in a process nominated in the assertion.

Remark 2.2 The above grammar does not have the operator of matching (often considered in the π-calculus), and uses replication rather than recursive definitions of agents. We have omitted matching not because of difficulties in its treatment, but because we think that causal bisimulation has interesting congruence properties in the calculus without matching. A discussion on this is deferred to the concluding section. We adopted replication because it suffices to code up recursion (the encoding showed by Milner [Mil91, section 3.1] for the ordinary interleaving bisimilarity also works well for causal bisimulation) and is easier to handle algebraically. $\square$

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*, 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*, are just tuples 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, a prefix $a(\tilde{b}).\, P$ or $\overline{a}\langle\tilde{b}\rangle.\, P$ is *well-sorted* for a sorting Ob if $a : s$ and $s \mapsto (\tilde{s}) \in Ob$ imply $\tilde{b} : \tilde{s}$. A process P is *well-sorted* for Ob if all input and output prefixes in P are well-sorted. The sorting system also affects substitutions, which are only defined between names of the same sort. In the remainder of the paper, all processes are supposed to be well-sorted for some sorting Ob.

2.3 The standard transition system and observation equivalence

We now come to the definition of observation equivalence. In an interleaving semantics, a π-calculus process has three possible forms of action. A *silent action* $P \xrightarrow{\tau} P'$ represents interaction, i.e. an internal activity in P. *Input* and *output actions* are, respectively, of the form

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

In both cases, the action occurs at a. In the input action, $\tilde{b}$ is the tuple of names which are received. In the output action, $\tilde{b}$ is the tuple of names which are emitted, and $\tilde{b'} \subseteq \tilde{b}$

are private names which are carried out from their current scope. The angled brackets in the input action are to stress that names $\widetilde{b}$ represent the *values* communicated — similarly to their role in an output action $\overline{a}\langle\widetilde{b}\rangle$ — as opposed to *binders*, which are found in an input prefix $a(\widetilde{b}).\,P$.

We use μ to represent the label of a generic action (not to be confused with α, which represents 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 $bn(\mu) \cup fn(\mu)$. Table 1 shows the *standard transition system* of the π-calculus.[1] 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.* We often abbreviate $P \xrightarrow{\tau} Q$ with $P \longrightarrow Q$, and write $P \xrightarrow{\widehat{\mu}} Q$ to mean $P \xrightarrow{\mu} Q$, if $\mu \neq \tau$, and $P = Q$ or $P \xrightarrow{\tau} Q$, if $\mu = \tau$. We write the composition of two relations $\mathcal{R}$ and $\mathcal{R}'$ as $\mathcal{R}\,\mathcal{R}'$.

Definition 2.3 (strong bisimilarity) *A symmetric relation* $\mathcal{R} \subseteq \mathcal{P}(\mathcal{N}) \times \mathcal{P}(\mathcal{N})$ *is a* strong bisimulation *if* $P\,\mathcal{R}\,Q$ *and* $P \xrightarrow{\mu} P'$ *imply that*[2] *there exists* Q' *s.t.* $Q \xrightarrow{\mu} Q'$ *and* $P'\,\mathcal{R}\,Q'$. *Two processes* P *and* Q *are* strongly bisimilar *written* $P \sim Q$, *if* $P\,\mathcal{R}\,Q$, *for some strong bisimulation* $\mathcal{R}$.

Relation $\sim$ is preserved by all π-calculus operators but input prefix, for which, however, a special inference rule is valid:

$$\text{if, for each } \widetilde{c},\ P\{\widetilde{c}/\widetilde{b}\} \sim Q\{\widetilde{c}/\widetilde{b}\} \text{ then } a(\widetilde{b}).\,P \sim a(\widetilde{b}).\,Q.$$

We sometimes also use the largest *congruence* contained in $\sim$, denoted by $\sim^{\mathrm{c}}$, and defined as follows:

$$P \sim^{\mathrm{c}} Q \text{ if and only if, for each name substitution } \sigma,\ P\sigma \sim Q\sigma.$$

---

[1] 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.* See [PS93, San93], for instance, for discussions about different forms of transition system and bisimulation for the π-calculus. In this paper we only consider the early system.

[2] We omit the requirement $bn(\mu) \cap fn(Q) = \emptyset$, since we work up-to alpha conversion.

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

$$\texttt{S-TAU:}\quad \tau.\,P \xrightarrow{\tau} P \qquad\qquad \texttt{S-PAR:}\quad \frac{P \xrightarrow{\mu} P'}{P \mid Q \xrightarrow{\mu} P' \mid Q}\quad bn(\mu) \cap fn(Q) = \emptyset$$

$$\texttt{S-COM:}\quad \frac{P \xrightarrow{(\boldsymbol{\nu}\,\widetilde{b'})\overline{a}\langle\widetilde{b}\rangle} P' \qquad Q \xrightarrow{a\langle\widetilde{b}\rangle} Q'}{P \mid Q \xrightarrow{\tau} \boldsymbol{\nu}\,\widetilde{b'}\,(P' \mid Q')}\quad \widetilde{b'} \cap fn(Q) = \emptyset$$

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

$$\texttt{S-SUM:}\quad \frac{P \xrightarrow{\mu} P'}{P + Q \xrightarrow{\mu} P'} \qquad\qquad \texttt{S-REP:}\quad \frac{P \mid\, !\,P \xrightarrow{\mu} P'}{!\,P \xrightarrow{\mu} P'}$$

Table 1: SOS for the standard processes

Table 2 lists a few simple algebraic laws which are valid for $\sim^{c}$ (and hence for the relations coarser than it, such as $\sim$). For $\sim$, an expansion law similar to that used in CCS [Mil89] is valid; details can be found in [MPW92] and [PS93].

Remark 2.4 The third law for restriction in Table 2 allows us to extrude the scope of a restriction across a parallel composition, provided that the restricted name is fresh. Since we work up-to alpha conversion, we shall apply the law without recalling this side condition. $\square$

As usual, the 'weak' arrow $\Longrightarrow$ is the reflexive and transitive closure of $\longrightarrow$, that is $\Longrightarrow = \bigcup_{n\geq 0} \longrightarrow^{n}$; then $\overset{\mu}{\Longrightarrow}$ stands for $\Longrightarrow\xrightarrow{\mu}\Longrightarrow$. If $P \Longrightarrow Q$ then we say that Q is a *τ-derivative* of P. Finally, similarly to $P \xrightarrow{\widehat{\mu}} Q$, we use $P \overset{\widehat{\mu}}{\Longrightarrow} Q$ to mean: $P \overset{\mu}{\Longrightarrow} Q$, if $\mu \neq \tau$, and $P \Longrightarrow Q$, if $\mu = \tau$.

On the weak arrows we define observation equivalence, sometimes called weak bisimulation. This is the interleaving behavioural equivalence we are most interested in.

Definition 2.5 (observation equivalence) *A symmetric relation $\mathcal{R} \subseteq \mathcal{P}(\mathcal{N}) \times \mathcal{P}(\mathcal{N})$ is a* weak bisimulation *(or a* $\approx$-bisimulation*) if $P\,\mathcal{R}\,Q$ and $P \overset{\mu}{\Longrightarrow} P'$ imply that there exists Q' s.t. $Q \overset{\widehat{\mu}}{\Longrightarrow} Q'$ and $P'\,\mathcal{R}\,Q'$. Two processes P and Q are* observationally equivalent, *(or* weakly bisimilar*), written $P \approx Q$, if $P\,\mathcal{R}\,Q$, for some weak bisimulation $\mathcal{R}$.*

Abelian monoid laws for $+$:	$P_1 + P_2$	$\sim^{c}$	$P_2 + P_1$
	$P_1 + (P_2 + P_3)$	$\sim^{c}$	$(P_1 + P_2) + P_3$
	$P + \mathbf{0}$	$\sim^{c}$	P
Abelian monoid laws for $\mid$:	$P_1 \mid P_2$	$\sim^{c}$	$P_2 \mid P_1$
	$P_1 \mid (P_2 \mid P_3)$	$\sim^{c}$	$(P_1 \mid P_2) \mid P_3$
	$P \mid \mathbf{0}$	$\sim^{c}$	P
Restriction:	$\boldsymbol{\nu}\, a\, \mathbf{0}$	$\sim^{c}$	$\mathbf{0}$
	$\boldsymbol{\nu}\, a\, \boldsymbol{\nu}\, b\, P$	$\sim^{c}$	$\boldsymbol{\nu}\, b\, \boldsymbol{\nu}\, a\, P$
if $a \notin fn(P_2)$, then	$(\boldsymbol{\nu}\, a\, P_1) \mid P_2$	$\sim^{c}$	$\boldsymbol{\nu}\, a\, (P_1 \mid P_2)$
Replication:	$!\, P$	$\sim^{c}$	$P \mid !\, P$

Table 2: A few laws for strong bisimulation congruence ($\sim^{c}$)

Observation equivalence is preserved by all operators except input prefix and sum; for input prefix, a congruence rule similar to that stated for $\sim$ is valid.

3 Proof techniques

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

3.1 The expansion relation

The expansion relation $\lesssim$ [AKH92, SM92] 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. The expansion relation provides us with better 'control' on τ-moves than the ordinary $\approx$.

Definition 3.1 (expansion) *A relation $\mathcal{R}$ 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 $\mathcal{R}$.*

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. Relation $\lesssim$, which is indeed a preorder, enjoys the same congruence properties as $\approx$. The following proposition sums up the relationship among the behavioural relations considered in the paper.

Proposition 3.2 *The following is a chain of strict inclusions: $\sim^{\mathrm{c}} \subset \sim \subset \lesssim \subset \approx$.* $\square$

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 [Mil89, SM92, San94a]. The up-to techniques which we use in the paper allow us to manipulate the derivatives of a pair of processes in two ways: First, we can replace such derivatives with behaviourally-equivalent processes; secondly, we can cut common contexts of the derivaitves. The contexts we cut are of the form $\boldsymbol{\nu}\, \widetilde{c}\, (R \mid [\cdot])$; they are called *static contexts.*

Definition 3.3 ($\approx$-bisimulation up-to context and up-to $\gtrsim$) *A symmetric relation $\mathcal{R}$ is a* $\approx$-bisimulation up-to context and up-to $\gtrsim$ *if $P \mathcal{R} Q$ and $P \overset{\mu}{\Longrightarrow} P''$ imply that there are a static context $C[\cdot]$ and processes P' and Q' s.t. $P'' \gtrsim C[P']$, $Q \overset{\widehat{\mu}}{\Longrightarrow} \gtrsim C[Q']$ and $P' \mathcal{R} Q'$.*

Proposition 3.4 *If $\mathcal{R}$ is a $\approx$-bisimulation up-to context and up-to $\gtrsim$, then $\mathcal{R} \subseteq \approx$.*

Proof: By showing that

$$\begin{array}{rll} \mathcal{R}' = \{(P, Q) & : & \text{for some static context } C[\cdot] \text{ and processes } P', Q' \\ & & \text{it holds that } P \gtrsim C[P'], Q \gtrsim C[Q'] \text{ and } P' \mathcal{R} Q' \quad \} \end{array}$$

is a $\approx$-bisimulation. This proves the thesis because $\mathcal{R} \subseteq \mathcal{R}'$. Details of the proof can be found in [San94b]. $\square$

We do not know whether the above soundness theorem holds if Definition 3.3 is weakened by replacing all occurrences of $\lesssim$ with $\approx$.

The next up-to technique is new; it is the corresponding for expansion of Definition 3.3. Note the occurrence of $\sim$ in clause (1) of Definition 3.5: We do not know whether the replacement of $\sim$ with $\lesssim$ preserves the soundness of the technique.

Definition 3.5 (expansion up-to context and up-to $\gtrsim$) *A relation $\mathcal{R}$ is an* expansion up-to context and up-to $\gtrsim$ *if $P \mathcal{R} Q$ implies:*

1. *Whenever $P \xrightarrow{\mu} P''$, there are a static context $C[\cdot]$ and processes P' and Q' s.t. $P'' \sim C[P']$, $Q \overset{\mu}{\Longrightarrow} \gtrsim C[Q']$ and $P' \mathcal{R} Q'$;*
2. *whenever $Q \xrightarrow{\mu} Q''$, there are a static context $C[\cdot]$ and processes P' and Q' s.t. $Q'' \gtrsim C[Q']$, $P \xrightarrow{\widehat{\mu}} \lesssim C[P']$ and $P' \mathcal{R} Q'$.*

Proposition 3.6 *If $\mathcal{R}$ is an expansion up-to context and up-to $\gtrsim$, then $\mathcal{R} \subseteq \lesssim$.*

PROOF: By showing that

$$\begin{array}{ll} \{(P,Q)\ : & \text{for some static context } C[\cdot] \text{ and processes } P', Q' \\ & \text{it holds that } P \lesssim C[P'], Q \gtrsim C[Q'] \text{ and } P' \mathcal{R} Q' \quad \} \end{array}$$

is an expansion relation. The proof is similar to the proof of Proposition 3.4. □

4 Causal bisimulation

The section begins with a discussion on the way object and subject dependencies in processes can be detected. Whereas the former are detectable with the standard transition system, the latter require a richer system, with explicit causes. We show a few properties of the richer system and define causal bisimulation on top of it.

Object dependencies are the dependencies determined by the binding mechanisms on names. To know the object dependencies in a process, we look at its *runs*. A run of a process is a sequence of transitions starting from that process.

Definition 4.1 *A run $P_1 \xrightarrow{\mu_1} P_2 \ldots P_n \xrightarrow{\mu_n} P_{n+1}$, $n \geq 1$, is* fresh *when for all $1 \leq i \leq n$ if $\mu_i = \boldsymbol{\nu}\, \widetilde{b}\, \overline{a}\langle\widetilde{c}\rangle$ or $\mu_i = a(\widetilde{b})$, then $\widetilde{b} \cap fn(P_i) = \emptyset$ and for all $j < i$, $\widetilde{b} \cap n(\mu_j) = \emptyset$; in this case, we say that names $\widetilde{b}$ are* introduced in *μ_i.*

In a fresh run, names exported in an output or imported in an input are all fresh. Fresh runs reveal all possible name dependencies in processes in a simple way.

Definition 4.2 *Given a fresh run $P_1 \xrightarrow{\mu_1} P_2 \ldots P_n \xrightarrow{\mu_n} P_{n+1}$, we say that μ_i is* object dependent on *μ_j, $1 \leq j < i \leq n$, if there is a name introduced in μ_j which is among the free names of μ_i.*

For instance, in the run $\boldsymbol{\nu}\, b\,(b \mid a(x).\, \overline{x}\langle b\rangle) \xrightarrow{a\langle c\rangle} \xrightarrow{\boldsymbol{\nu}\, b\, \overline{c}\langle b\rangle} \xrightarrow{b} \mathbf{0} \mid \mathbf{0}$, the second action is object dependent on the first and the third is object dependent on the second. Object dependencies can already be observed at the interleaving level, since they are shown by the occurrences of names in actions: Names themselves act as pointers among actions. Thus, observationally equivalent processes exhibit the same object dependencies.

Proposition 4.3 *If* $P_1 \approx Q_1$ *and* P_1 *has a fresh run* $P_1 \xrightarrow{\mu_1} P_2 \ldots P_n \xrightarrow{\mu_n} P_{n+1}$ *where* μ_i *is object dependent on* μ_j*, then also* Q_1 *has a fresh run* $Q_1 \xrightarrow{\mu_1} Q_2 \ldots Q_n \xrightarrow{\mu_n} Q_{n+1}$ *where* μ_i *is object dependent on* μ_j*.* $\square$

The other form of dependencies among actions, namely the *subject dependencies*, are determined by the nesting of prefixes. By contrast with the object dependencies, the subject dependencies are not detected in an interleaving semantics (see discussion in Section 1). To capture them, we add explicit causal information to the standard syntax and operational semantics of the calculus. For this, we follow Kiehn's system [Kie91], which describes the causal dependencies for CCS processes, and transport it onto the π-calculus.

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 is revealed by means of the set $K \subseteq_{\mathrm{fin}} \mathcal{K}$ of their associated causes. We call K the *cause-set* for μ. If a successive action μ' is caused by μ, then μ' will have k (i.e., the cause associated with μ) in its cause-set. In the term syntax, the explicit use of causes is accompanied by the introduction of a causal prefix $K :: A$, for $K \subseteq_{\mathrm{fin}} \mathcal{K}$, which says that the cause-set 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

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

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 cause-sets $\{k_1\}$ and $\{k_2\}$ of the consumed actions are merged.

4.1 Syntax and transitional semantics of causal terms

The notations and conventions introduced on standard processes — including the identification of alpha equivalent processes — extend to the enriched language of causal processes below, and will not be repeated.

We let k and h range over causes (i.e., elements of $\mathcal{K}$), K and H over finite subsets of $\mathcal{K}$, and A and B over causal processes. We denote by $\mathcal{K}(A)$ the set of causes which appear in the causal process A. To improve readability, in the syntax of our terms and of our operational rules we abbreviate a union of sets of causes $K \cup K'$ as K, K' and we write a singleton set $\{k\}$ simply as k. Thus, $A \xrightarrow[K,K';k]{\mu} A'$ stands for $A \xrightarrow[K \cup K';k]{\mu} A'$, and $k :: A$ stands for $\{k\} :: A$. [1]

Definition 4.4 (causal processes) *Consider an infinite countable set* $\mathcal{N}$ *of names, and an infinite countable set* $\mathcal{K}$ *of* causes *with* $\mathcal{K}$, $\mathcal{N}$ *and* $\{\tau\}$ *pairwise disjoint. The language of* polyadic causal processes over causes $\mathcal{K}$ and names $\mathcal{N}$, *written* $\mathcal{P}_{\mathrm{c}}(\mathcal{N}, \mathcal{K})$, *is given by the following grammar:*

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

where P *is a* $\mathcal{P}(\mathcal{N})$ *term (i.e., a standard term) and* $K \subseteq_{\mathrm{fin}} \mathcal{K}$*.*

We do 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. This is clear from the transition rules for causal terms, in Table 3. The rules for the prefixes and for communication express the essence of the whole semantics: INP and OUT, for input and output prefixes, introduce causes into processes; CAU, for the causal prefix $K :: A$, makes any action performed by A dependent upon K. Rule COM implements the cause exchange discussed at the beginning of this section. In the rule, the notation $A[k \rightsquigarrow K]$ indicates the replacement of k with the elements of K, in any set of causes in A. For instance, $(\{k_1, k_2\} :: \alpha.\mathbf{0})[k_2 \rightsquigarrow \{k_3, k_4\}]$ is $\{k_1, k_3, k_4\} :: \alpha.\mathbf{0}$. The rules for the remaining operators are formally identical to the standard rules of Table 1.

Our rules, when restricted to the sublanguage of finite 0-adic (i.e. CCS) causal terms, coincide with Kiehn's rules [Kie91]. The only variation is in rule COM: In our case, processes

---

[1] The prefix $k :: A$ is taken as a primitive operator in [Kie91].

Rules for visible transitions:

$$\mathtt{Out}:\ \overline{a}\langle\widetilde{b}\rangle.\,A \xrightarrow[\emptyset\,;\,k]{\overline{a}\langle\widetilde{b}\rangle} k :: A \qquad \mathtt{Inp}:\ a(\widetilde{b}).\,A \xrightarrow[\emptyset\,;\,l]{a\langle\widetilde{c}\rangle} k :: A\{\widetilde{c}/\widetilde{b}\}$$

$$\mathtt{Cau}:\ \frac{A \xrightarrow[K;k]{\mu} A'}{K' :: A \xrightarrow[K,K';k]{\mu} K' :: A'} \qquad \mathtt{Par}:\ \frac{A_1 \xrightarrow[K;k]{\mu} A_1'}{A_1 \mid A_2 \xrightarrow[K;k]{\mu} A_1' \mid A_2}\, bn(\mu) \cap fn(A_2) = \emptyset$$

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

$$\mathtt{Sum}:\ \frac{A_1 \xrightarrow[K;k]{\mu} A_1'}{A_1 + A_2 \xrightarrow[K;k]{\mu} A_1'} \qquad \mathtt{Rep}:\ \frac{A \mid !\,A \xrightarrow[K;k]{\mu} A'}{!\,A \xrightarrow[K;k]{\mu} A'}$$

……

Rules for silent transitions:

$$\mathtt{T\text{-}pre}:\ \tau.A \xrightarrow{\tau} A \qquad \mathtt{T\text{-}sum}:\ \frac{A_1 \xrightarrow{\tau} A_1'}{A_1 + A_2 \xrightarrow{\tau} A_1'}$$

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

$$\mathtt{Com}:\ \frac{A_1 \xrightarrow[K_1;k]{(\boldsymbol{\nu}\,\widetilde{b'})\overline{a}\langle\widetilde{b}\rangle} A_1' \qquad A_2 \xrightarrow[K_2;k]{a\langle\widetilde{b}\rangle} A_2'}{A_1 \mid A_2 \xrightarrow{\tau} \boldsymbol{\nu}\,\widetilde{b'}\,(A_1'[k \rightsquigarrow K_2] \mid A_2'[k \rightsquigarrow K_1])}\ \widetilde{b'} \cap fn(A_2) = \emptyset\,,\ k \notin \mathcal{K}(A_1, A_2)$$

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

Table 3: SOS of causal terms

A_1 and A_2 use the same fresh cause k, whereas the obvious π-calculus's adaptation of Kiehn's rule, called `Com'` below, uses two distinct causes:

$$\texttt{Com}' : \quad \frac{A_1 \xrightarrow[K_1;k_1]{(\boldsymbol{\nu}\widetilde{b}')\overline{a}\langle\widetilde{b}\rangle} A_1' \quad A_2 \xrightarrow[K_2;k_2]{a\langle\widetilde{b}\rangle} A_2'}{A_1 \mid A_2 \xrightarrow{\tau} \boldsymbol{\nu}\widetilde{b}'\,(A_1'[k_1 \rightsquigarrow K_2] \mid A_2'[k_2 \rightsquigarrow K_1])} \quad k_1 \notin \mathcal{K}(A_1), k_2 \notin \mathcal{K}(A_2), \widetilde{b}' \cap fn(A_2) = \emptyset\,.$$

Anyhow, the two rules `Com` and `Com'` are interderivable, as can be proved using Lemma 4.5 below to rename causes.

Weak causal transitions are defined in the usual manner, therefore $A \overset{\mu}{\underset{K;k}{\Longrightarrow}} A'$ is $A \Longrightarrow \xrightarrow[K;k]{\mu} \Longrightarrow A'$. The definition of well-sortedness for causal terms is given as for standard processes (all input and output prefixes in a term must respect the discipline imposed by the chosen sorting). All causal terms considered in the paper are supposed to be well-sorted.

4.2 Basic Properties of the Transitional Semantics of Causal Terms

A *cause substitution* ρ is a function from $\mathcal{K}$ to $\mathcal{K}$. We write $k\rho = k'$ if ρ maps cause k onto k'; and $K\rho$ for the set of causes $\{k\rho \ : \ k \in K\}$. Also, $A\rho$ is the process obtained from A by replacing each of its causes k with $k\rho$. The cause substitution that maps k onto k_0 and is the identity elsewhere, will be sometimes written as a replacement $[k \rightsquigarrow k_0]$.

Lemma 4.5 allows us to rename the cause k of a transitions $A \xrightarrow[K;k]{\mu} A'$. Later, Lemma 4.8 will extend this result to weak transitions. Lemmata 4.6 and 4.7 relate the actions of causal terms A and $A\rho$.

Lemma 4.5 *Let $k, k_0 \in \mathcal{K}$, with $k_0 \notin \mathcal{K}(A)$.*

1. *If $A \xrightarrow[K;k]{\mu} A'$, then $A \xrightarrow[K;k_0]{\mu} A''$ and $A''[k_0 \rightsquigarrow k] = A'$.*
2. *If $A \xrightarrow[K;k_0]{\mu} A'$, then $A \xrightarrow[K;k]{\mu} A''$ and $A'[k_0 \rightsquigarrow k] = A''$.*

PROOF: Easy transition inductions. □

Lemma 4.6 *Let ρ be a cause substitution. Then*

1. *$A \xrightarrow[K;k]{\mu} A'$ implies $A\rho \xrightarrow[K\rho;k\rho]{\mu} A'\rho$.*
2. *$A \xrightarrow{\tau} A'$ implies $A\rho \xrightarrow{\tau} A'\rho$.*

3. *$A \Longrightarrow A'$ implies $A\rho \Longrightarrow A'\rho$.*

4. *$A \underset{K;k}{\overset{\mu}{\Longrightarrow}} A'$ implies $A\rho \underset{K\rho;k\rho}{\overset{\mu}{\Longrightarrow}} A'\rho$.*

Proof: (1) is proved by transition induction; (2) is also proved by transition induction, but in the case when `Com` is the last rule applied, we also need part (1) of this lemma and Lemma 4.5. Assertion (3) is proven by iterating (2) and finally (4) is a consequence of (1) and (3). □

Lemma 4.7 *Let ρ be a cause substitution. Then*

1. *If $A\rho \underset{K;k}{\overset{\mu}{\longrightarrow}} A'$ then there are K', k' and A'' s.t. $A \underset{K';k'}{\overset{\mu}{\longrightarrow}} A''$ with $K'\rho = K$, $k'\rho = k$ and $A''\rho = A'$.*

2. *If $A\rho \xrightarrow{\tau} A'$ then there is A'' s.t. $A \xrightarrow{\tau} A''$ with $A''\rho = A'$.*

3. *If $A\rho \Longrightarrow A'$ then there is A'' s.t. $A \Longrightarrow A''$ with $A''\rho = A'$.*

4. *If $A\rho \underset{K;k}{\overset{\mu}{\Longrightarrow}} A'$ then there are K', k' and A'' s.t. $A \underset{K';k'}{\overset{\mu}{\Longrightarrow}} A''$ with $K'\rho = K$, $k'\rho = k$ and $A''\rho = A'$.*

Proof: Similar to that of Lemma 4.6. □

By combining Lemmata 4.5, 4.6(3) and 4.7(3), we obtain the analogue of Lemma 4.5 for weak transitions:

Lemma 4.8 *Let $k, k_0 \in \mathcal{K}$, with $k_0 \notin \mathcal{K}(A)$.*

1. *If $A \underset{K;k}{\overset{\mu}{\Longrightarrow}} A'$, then $A \underset{K;k_0}{\overset{\mu}{\Longrightarrow}} A''$ and $A''[k_0 \leadsto k] = A'$.*

2. *If $A \underset{K;k_0}{\overset{\mu}{\Longrightarrow}} A'$, then $A \underset{K;k}{\overset{\mu}{\Longrightarrow}} A''$ and $A'[k_0 \leadsto k] = A''$.* □

4.3 Causal bisimulation

Weak causal bisimulation is defined on top of weak causal transitions by means of familiar bisimulation techniques. We shall omit the adjective "weak" and simply call it *causal bisimulation.*

Definition 4.9 (causal bisimulation) *A symmetric relation $\mathcal{R} \subseteq \mathcal{P}_\mathrm{c}(\mathcal{N},\mathcal{K}) \times \mathcal{P}_\mathrm{c}(\mathcal{N},\mathcal{K})$ is a* causal bisimulation *(or $\approx_\mathrm{c}$-*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* $A \overset{\mu}{\underset{K;k}{\Longrightarrow}} A'$ *with* $k \notin \mathcal{K}(A,B)$*, there exists* B' *s.t.* $B \overset{\mu}{\underset{K;k}{\Longrightarrow}} B'$ *and* $A' \mathcal{R} B'$*.*

Two causal terms A, B *are* causally bisimilar*, written* $A \approx_{\mathrm{c}} B$*, if* $A \mathcal{R} B$ *for some causal bisimulation* $\mathcal{R}$*.* □

Lemma 4.10 *For any cause substitution* ρ*, if* $A \approx_{\mathrm{c}} B$ *then* $A\rho \approx_{\mathrm{c}} B\rho$*.*

Proof: We show that

$$\mathcal{R} = \{(A\rho, B\rho) \ : \ \rho \text{ is a cause substitution and } A \approx_{\mathrm{c}} B\}$$

is a $\approx_{\mathrm{c}}$-bisimulation. Suppose $A \approx_{\mathrm{c}} B$, and consider clause (2) of Definition 4.9. If

$$A\rho \overset{\mu}{\underset{K;k}{\Longrightarrow}} A' \text{ with } k \notin \mathcal{K}(A\rho) \tag{1}$$

then from Lemma 4.7, for some K', k' and A'', we have

$$A \overset{\mu}{\underset{K';k'}{\Longrightarrow}} A'' \quad \text{with } A''\rho = A',\ K'\rho = K \text{ and } k'\rho = k. \tag{2}$$

Let $k_0 \notin \mathcal{K}(A,B)$; by Lemma 4.8,

$$A \overset{\mu}{\underset{K';k_0}{\Longrightarrow}} A''' \quad \text{with } A'''[k_0 \rightsquigarrow k'] = A''$$

and, since $A \approx_{\mathrm{c}} B$, also

$$B \overset{\mu}{\underset{K';k_0}{\Longrightarrow}} B''' \quad \text{with } A''' \approx_{\mathrm{c}} B'''.$$

Reversing, the steps, we get

$$B \overset{\mu}{\underset{K';k'}{\Longrightarrow}} B'' \quad \text{with } B'''[k_0 \rightsquigarrow k'] = B''$$

and

$$B\rho \overset{\mu}{\underset{K'';k''}{\Longrightarrow}} B' \quad \text{with } K'' = K'\rho,\ k'' = k'\rho \text{ and } B' = B''\rho\,. \tag{3}$$

The action in (3) matches the one in (1): The conditions in (2) and (3) imply $K = K''$ and $k'' = k$; moreover, it holds that $B' = B'''[k_0 \rightsquigarrow k']\rho$, $A' = A'''[k_0 \rightsquigarrow k']\rho$ and $A''' \approx_{\mathrm{c}} B'''$. We conclude that $(A', B') \in \mathcal{R}$.

The other clause of the definition of $\approx_{\mathrm{c}}$ can be dealt with similarly. □

The following is a characterisation of causal bisimulation which will be exploited in the proof of completeness for the encoding. The difference from Definition 4.9 is that the requirement $k \notin \mathcal{K}(A,B)$ in clause (2), input case, is dropped.

Definition 4.11 *A symmetric relation* $\mathcal{R} \subseteq \mathcal{P}_{\mathrm{c}}(\mathcal{N},\mathcal{K}) \times \mathcal{P}_{\mathrm{c}}(\mathcal{N},\mathcal{K})$ *is a* loose causal bisimulation *(or* $\approx'_{\mathrm{c}}$-bisimulation*) if* $A\,\mathcal{R}\,B$ *implies:*

1. *whenever* $A \Longrightarrow A'$ *there exists* B *s.t.* $B \Longrightarrow B'$ *and* $(A', B') \in \mathcal{R}$*;*
2. *whenever* $A \overset{a\langle\widetilde{b}\rangle}{\underset{K;k}{\Longrightarrow}} A'$ *there exists* B' *s.t.* $B \overset{a\langle\widetilde{b}\rangle}{\underset{K;k}{\Longrightarrow}} B'$ *and* $(A', B') \in \mathcal{R}$*;*
3. *whenever* $A \overset{\nu\widetilde{b}\,\overline{a}\langle\widetilde{b}\rangle}{\underset{K;k}{\Longrightarrow}} A'$ *with* $k \notin \mathcal{K}(A, B)$*, there exists* B' *s.t.* $B \overset{\nu\widetilde{b}\,\overline{a}\langle\widetilde{b}\rangle}{\underset{K;k}{\Longrightarrow}} B'$ *and* $(A', B') \in \mathcal{R}$*.*

Two causal terms A *and* B *are* loose cause bisimilar*, written* $A \approx'_{\mathrm{c}} B$*, if* $A\,\mathcal{R}\,B$*, for some loose causal bisimulation* $\mathcal{R}$*.*

Proposition 4.12 *The relations* $\approx_{\mathrm{c}}$ *and* $\approx'_{\mathrm{c}}$ *coincide.*

PROOF: We only have to show that $\approx_{\mathrm{c}} \subseteq \approx'_{\mathrm{c}}$; the converse follows immediately because the defining clauses for $\approx'_{\mathrm{c}}$ are equal or stronger (in the input case) than those for $\approx_{\mathrm{c}}$.

We proof that $\approx_{\mathrm{c}}$ is a $\approx'_{\mathrm{c}}$-bisimulation. Let $A \approx_{\mathrm{c}} B$. We check clause (2) of Definition 4.11. Suppose $A \overset{a\langle\widetilde{b}\rangle}{\underset{K;k}{\Longrightarrow}} A'$, and take a fresh cause k_0. Then from Lemma 4.8, letting $\mu = a\langle\widetilde{b}\rangle$,

$$A \overset{\mu}{\underset{K;k_0}{\Longrightarrow}} A'', \quad \text{with } A''[k_0 \rightsquigarrow k] = A',$$

which implies, since $A \approx_{\mathrm{c}} B$,

$$B \overset{\mu}{\underset{K;k_0}{\Longrightarrow}} B'', \quad \text{with } A'' \approx_{\mathrm{c}} B''.$$

Again, by Lemma 4.8, we get

$$B \overset{\mu}{\underset{K;k}{\Longrightarrow}} B', \quad \text{with } B''[k_0 \rightsquigarrow k] = B'.$$

From Lemma 4.10, $A'' \approx_{\mathrm{c}} B''$ implies $A' = A''[k_0 \rightsquigarrow k] \approx_{\mathrm{c}} B''[k_0 \rightsquigarrow k] = B'$, which concludes the case. □

5 The Encoding

In this section we present and discuss the encoding of causal bisimulation into observation equivalence.

We use two new sorts $\mathcal{C}$ and $\mathcal{T}$ of names, 'new' meaning that these sorts do not occur in the original sorting of causal terms. There will be a one-to-one correspondence between causes — i.e. elements of the set $\mathcal{K}$ — in a causal process, and names of sort $\mathcal{C}$ in the standard process enconding it. Hence, for notational convenience, we identify the sets $\mathcal{C}$ and $\mathcal{K}$ (any ambiguity is resolved by the different classes of processes in which elements of these sets occur as causes and as names, and by the different use of causes and names), and call a name in $\mathcal{C}$ a *cause*, or also a *cause name*. Names in $\mathcal{C}$ carry names of sort $\mathcal{T}$. We sometimes call a name in $\mathcal{T}$ a *token*.

The intuition underlying the encoding is as follows. Causes in a causal term A are encoded in the standard term $[\![A]\!]$ as standard processes called *wires*. A wire $k \rhd K$ takes a token in input at the *entrance point* k and emits it, as an output, at each *end-point* $k' \in K$. Two wires are connected when an end-point of the first is the entrance-point of the other; this connection represents union of causes. When $[\![A]\!]$ performs a visible action, a name k is emitted and a wire with entrance-point k is created. Intuitively, k represents the unique cause associated to that transition, whose cause-set is represented by all end-points of all wires reachable from k. An observer that receives k can determine this cause-set by sending a token at k and and observing *where* (i.e., at which end-points) the token can be retrieved.

As an example, the encoding of the causal process $K :: a.b.\mathbf{0}$ will have the following transitions:

$$[\![K :: a.b.\mathbf{0}]\!] \xrightarrow{a\langle k\rangle} k \rhd K \mid [\![k :: b.\mathbf{0}]\!] \xrightarrow{b\langle h\rangle} k \rhd K \mid h \rhd k \mid [\![h :: \mathbf{0}]\!] .$$

(This shows how *linear* chains are generated.) A crucial point is the modeling of silent transitions. The "merging of causes" discussed at the beginning of Section 4 is implemented as an exchange of a bound name, by which the appropriate wires get connected. As an example, we have:

$$[\![K :: a.P \mid H :: \overline{a}.Q]\!] \xrightarrow{\tau} \boldsymbol{\nu}\, k\,(\, k \rhd K \mid k \rhd H \mid [\![k :: P \mid k :: Q]\!])$$

(This shows how *branching* chains arise.) After the communication, P and Q have a common cause-set $K \cup H$, referred to by k. Indeed, the derivative of the above transition will turn out to be observationally equivalent to $[\![K \cup H :: P \mid K \cup H :: Q]\!]$.

We come to the actual definition of wire processes. In the sequel, we use v to range over $\mathcal{T}$, and we abbreviate $\mathcal{N} \cup \mathcal{C} \cup \mathcal{T}$ as $\mathcal{N}^{+}$.

Definition 5.1 (Wire processes) *The standard process* $k \rhd K \in \mathcal{P}(\mathcal{N}^+)$*, called* wire process*, is so defined:*

$$k \rhd K \overset{def}{=} !\, k(v).\, \Pi_{k' \in K}\, !\, \overline{k'}\langle v \rangle\,.$$

Name k *is the* entrance point *of the wire, and names in* K *are the* end-points*.*

We explain the presence of the replication operators in wire processes. In general, once a wire is created, an unbounded number of other wires may get connected to it, both at its entrance point, and — due to the branching structure of causal chains — at each of its end-points. The outermost replication in the definition of $k \rhd K$ is to make the wire *persistent*, so that it can be used an arbitrary number of times. The innermost replications in the definition of $k \rhd K$ make a token availability at an end-point persistent. This ensures that a token which has reached an end-point be transmitted to all unboundedly many wires connected with it.

The encoding is presented in Table 4. It acts as a homomorphism, from $\mathcal{P}_{\mathrm{c}}(\mathcal{N}, \mathcal{K})$ to $\mathcal{P}(\mathcal{N}^+)$, homomorphism everywhere but on prefixes. The encoding adds adds extra components (the wires) into processes, thus increasing the number of their states. The problem, however, is strongly alleviated by Lemmata 7.1 and 7.2 and the Cancelation Lemma 8.4, in Section 6, which allow us to get rid of wires when they appear at the outermost level of processes.

The encoding has to be defined on sortings too, since an extra component (of sort $\mathcal{C}$) is added to the arities of names in the target processes. To ease the notation, we abbreviate $k \rhd (K \cup H)$ as $k \rhd (K, H)$, and $k \rhd (\{k'\}, K)$ as $k \rhd (k', K)$. Similarly, we abbreviate $[\![A]\!](K \cup K')$ as $[\![A]\!](K, K')$ and $[\![A]\!](\{k\}, K')$ as $[\![A]\!](k, K')$. Note that when the source term are 0-adic terms (i.e., CCS terms), Table 4 gives us an encoding of the classical causal bisimulation of CCS [DD89, Kie91] into the observation equivalence of the monadic π-calculus. It is straightforward to see that there is agreement between the definitions of the encoding on processes and on sortings:

Proposition 5.2 *If* A *is well-sorted for* Ob *then* $[\![A]\!]$ *is well-sorted for* $[\![Ob]\!]$*.* $\square$

6 Fundamental Lemmata

All processes in this section are standard processes from $\mathcal{P}(\mathcal{N}^+)$.

Encoding of the sorting:

$$\begin{array}{rcl} \llbracket Ob \rrbracket & \stackrel{def}{=} & \{ s \mapsto (\tilde{s}, \mathcal{C}) \; : \; s \mapsto (\tilde{s}) \in Ob \} \; \bigcup \\ & & \{ \mathcal{C} \mapsto (\mathcal{T}), \; \mathcal{T} \mapsto () \} \end{array}$$

where $\mathcal{C}$ and $\mathcal{T}$ are new sorts, i.e. sorts which do not appear in Ob.

……

Encoding of causal processes:
We assume $k : \mathcal{C}$, $K \subseteq_{\mathrm{fin}} \mathcal{C}$ and $v : \mathcal{T}$; we set $\llbracket A \rrbracket \stackrel{def}{=} \llbracket A \rrbracket \emptyset$, where $\llbracket A \rrbracket K$ is defined by induction on the structure of A as follows:

$$\begin{array}{rcl@{\qquad}rcl} \llbracket a(\widetilde{b}).\, A \rrbracket K & \stackrel{def}{=} & \multicolumn{4}{l}{a(\widetilde{b}k).\,(\; k \rhd K \;\mid \llbracket A \rrbracket \{k\}),\; k \in \mathcal{K} - K} \\ \llbracket \overline{a}\langle \widetilde{b} \rangle.\, A \rrbracket K & \stackrel{def}{=} & \multicolumn{4}{l}{\boldsymbol{\nu}\, k\, \overline{a} \langle \widetilde{b}k \rangle.\,(\; k \rhd K \;\mid \llbracket A \rrbracket \{k\}),\; k \in \mathcal{K} - K} \\ \llbracket K' :: A \rrbracket K & \stackrel{def}{=} & \llbracket A \rrbracket (K', K) & \llbracket \tau.A \rrbracket K & \stackrel{def}{=} & \tau.\llbracket A \rrbracket K \\ \llbracket A_1 \mid A_2 \rrbracket K & \stackrel{def}{=} & \llbracket A_1 \rrbracket K \mid \llbracket A_2 \rrbracket K & \llbracket A_1 \, + \, A_2 \rrbracket K & \stackrel{def}{=} & \llbracket A_1 \rrbracket K \, + \, \llbracket A_2 \rrbracket K \\ \llbracket \boldsymbol{\nu}\, a\, A \rrbracket K & \stackrel{def}{=} & \boldsymbol{\nu}\, a\, \llbracket A \rrbracket K & \llbracket \mathbf{0} \rrbracket K & \stackrel{def}{=} & \mathbf{0} \\ \llbracket\, !\, A \rrbracket K & \stackrel{def}{=} & !\llbracket A \rrbracket K & & & \end{array}$$

Table 4: The encoding of causal terms

6.1 Properties of the replication operator

We recall some basic properties of the replication operator, which will be used several times in this and in the next section.

Lemma 6.1

1. $! P \mid ! P \sim^{c} ! P$.
2. $! \alpha.P \sim^{c} \alpha.(! \alpha.P \mid P)$, *if* $bn(\alpha) \cap fn(\alpha. P) = \emptyset$. □

Lemma 6.2 asserts the distributivity of restricted replications over parallel composition and replication. It generalises an original result by Milner [Mil91] to the case in which there is a *finite set* of restricted replications — rather than one only.

Lemma 6.2 *Suppose name k occurs free in the processes P_1, P_2, P and R_i ($i \in I$) only in output subject position. Then*

1. $\boldsymbol{\nu} k \left(P_1 \mid P_2 \mid \Pi_{i \in I} ! \, k(v). \, R_i\right) \;\sim^{c}\; \boldsymbol{\nu} k \left(P_1 \mid \Pi_{i \in I} ! \, k(v). \, R_i\right) \mid \boldsymbol{\nu} k \left(P_2 \mid \Pi_{i \in I} ! \, k(v). \, R_i\right)$.
2. $\boldsymbol{\nu} k \, (! \, P \mid \Pi_{i \in I} ! \, k(v). \, R_i) \;\sim^{c}\; ! \, \boldsymbol{\nu} k \, (P \mid \Pi_{i \in I} ! \, k(v). \, R_i)$.

Proof: Analogous to the proofs of Milner's distributivity laws in [Mil91]: One has to define strong bisimulations that are preserved by name substitutions. Some simplification can be achieved by working "up-to static contexts", as shown in [San94a]. □

Lemma 6.3

1. $\boldsymbol{\nu} k \, (! \, \alpha.P \mid \Pi_{i \in I} ! k(v).R_i) \sim \, ! \, \alpha. \, (\boldsymbol{\nu} k \, (P \mid \Pi_{i \in I} ! \, k(v). \, R_i))$,
   if $k \notin n(\alpha)$, $bn(\alpha) \cap fn(\Pi_{i \in I} ! \, k(v). \, R_i) = \emptyset$, *and k occurs free in P and R_i ($i \in I$) only in output subject position;*
2. $\boldsymbol{\nu} k \, (\Pi_{i \in I} ! k(v).\Pi_{k' \in K_i} ! \, \overline{k'} \langle v \rangle \mid ! \, \overline{k} \langle v \rangle) \gtrsim \Pi_{k' \in \cup_{i \in I} K_i} ! \, \overline{k'} \langle v \rangle$, *if* $k \notin \cup_{i \in I} K_i$.

Proof: (1) is proved by first using Lemma 6.2(2), to distribute the restricted replications $\Pi_{i \in I} ! k(v).R_i$ over the replication $! \, \alpha.P$, and then using expansion and restriction laws (Table 2), in the order; (2) is proved by exhibiting the appropriate expansion relation. □

6.2 Properties of wire-processes

Wire processes $k \rhd K$ are the core of our implementation of causal bisimulation. To prove the correctness of the implementation, we first need to establish a few properties of compositions of such wires (Lemmata 6.4 and 6.6). To begin with, two wires $k' \rhd (k, K')$ and $k \rhd K$, in which the entrance point of the second is among the end-points of the first, are, so to speak, 'connected': Hence the end-points K of the second can be added to the end-points K' of the first. This is, roughly, the content of Lemma 6.4.

If in addition to the above hypothesis, we assume that the common extreme k of the two wires is inaccessible, then their composition yields a single wire $k' \rhd (K, K')$, whose end-points include all end-points of the given wires, except the inaccessible k. This is, roughly, the content of Lemma 6.6 (the lemma is slightly more general, in that it allows a product $\Pi_{i \in I}\, k \rhd K_i$ of wires in place of the single wire $k \rhd K$.) The content of Lemmata 6.4 and 6.6 is depicted in Figure 1.

Lemma 6.4 $k' \rhd (k, K') \mid k \rhd K \;\gtrsim\; k' \rhd (k, K', K) \mid k \rhd K$.

Proof: Fixed K, K', k and k', let

$$\begin{aligned} P &\overset{def}{=} k \rhd K \mid k' \rhd (k, K', K) \\ Q &\overset{def}{=} k \rhd K \mid k' \rhd (k, K') \; . \end{aligned}$$

We prove that the singleton $\{(P, Q)\}$ is an expansion up to context and up to $\sim$ (This technique is just a particular case of the expansion up to context and up to $\gtrsim$ (Definition 3.5), since $\sim \subset \gtrsim$). We show how the moves of P are matched by Q; the converse is similar. The only non-trivial case is when the move of P originates from $k' \rhd (k, K', K)$, as follows:

$$P \xrightarrow{k'(w)} \sim \Pi_{k'' \in \{k\} \cup K' \cup K} \, ! \, \overline{k''}\langle w \rangle \mid P \, .$$

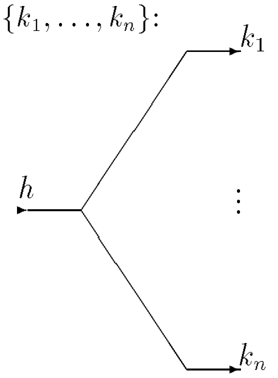
This can be matched by Q thus:

$$\begin{aligned} Q \quad \xrightarrow{k'(w)} \sim \quad & k \rhd K \mid \Pi_{k'' \in \{k\} \cup K'} \, ! \, \overline{k''}\langle w \rangle \mid k' \rhd (k, K') \\ \xrightarrow{\tau} \sim \quad & \Pi_{k'' \in K} \, ! \, \overline{k''}\langle w \rangle \mid \Pi_{k'' \in \{k\} \cup K'} \, ! \, \overline{k''}\langle w \rangle \mid Q \\ \sim \quad & \Pi_{k'' \in \{k\} \cup K' \cup K} \, ! \, \overline{k''}\langle w \rangle \mid Q \end{aligned}$$

where the last step, in the case when $K \cap (\{k\} \cup K') \neq \emptyset$, Lemma 6.1 is needed. This is enough because $\{(P, Q)\}$ is an expansion up-to context and up-to $\sim$. $\square$

To prove Lemma 6.6, we use the following lemma:

The wire process $h \triangleright K$, with $K = \{k_1, \ldots, k_n\}$:



The wire equations $h' \triangleright (H, h) \mid h \triangleright K \gtrsim h' \triangleright (K, H, h) \mid h \triangleright K$, and $(\boldsymbol{\nu}\, h)(\, h' \triangleright (H, h) \mid h \triangleright K\,) \gtrsim h' \triangleright (K, H)$, for $H = \{h_1, \ldots, h_n\}$, $K = \{k_1, \ldots, k_m\}$; in the latter, $h \notin K \cup K$:

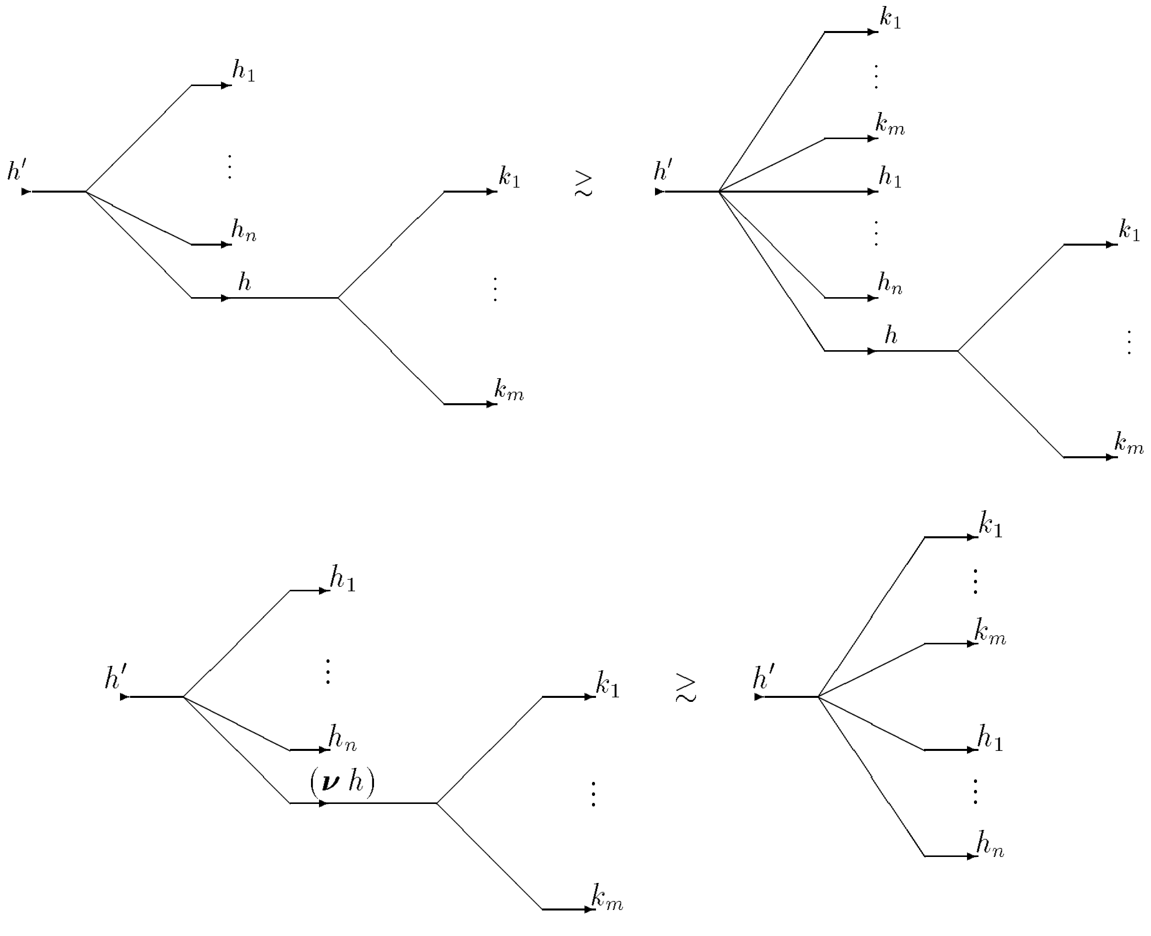


Figure 1: Graphical representations of wire processes.

Lemma 6.5

1. *$\boldsymbol{\nu} k\,(\,!\,\alpha.P \mid \Pi_{i\in I}\; k \rhd K_i\,) \sim\; !\,\alpha.(\boldsymbol{\nu} k\,(P \mid \Pi_{i\in I}\; k \rhd K_i\,))$, if $k \notin n(\alpha)$ and k can occur free in P only in output subject position;*

2. *$\boldsymbol{\nu} k\,(\Pi_{i\in I}\; k \rhd K_i \;\mid !\overline{k}\langle v\rangle) \gtrsim \Pi_{k'\in\cup_{i\in I}K_i}\,!\,\overline{k'}\langle v\rangle$, if $k \notin \cup_{i\in I}K_i$.*

Proof: The clauses of this lemma are instances of clauses (1) and (2), respectively, of Lemma 6.3. □

Lemma 6.6 *If $k \notin \big(\{k'\} \cup K' \cup (\cup_{i\in I}K_i)\big)$, then*

$$\boldsymbol{\nu} k\,\big(\; k' \rhd (K', k) \;\mid \Pi_{i\in I}\; k \rhd K_i\;\big) \gtrsim \;k' \rhd (K', \cup_{i\in I}K_i)\;.$$

Proof: Let $Q \overset{def}{=} \boldsymbol{\nu} k\,\big(\; k' \rhd (K', k) \;\mid \Pi_{i\in I}\; k \rhd K_i\;\big)$, $P \overset{def}{=}\; k' \rhd (K', \cup_{i\in I}K_i)$ and $R \overset{def}{=} \Pi_{k''\in K'}\,!\,\overline{k''}\langle v\rangle \mid !\,\overline{k}\langle v\rangle$. Then, recalling that $k' \rhd (K', k) \;= !\,k'(v).R$, we have:

$$\begin{array}{lll} Q & \sim & !\,k'(v).\boldsymbol{\nu} k\,(R \mid \Pi_{i\in I}\; k \rhd K_i\,) \qquad \text{Lemma 6.5 (1)} \\ & \sim & !\,k'(v).\big(\Pi_{k''\in K'}\,!\,\overline{k''}\langle v\rangle \mid \boldsymbol{\nu} k\,(\,!\,\overline{k}\langle v\rangle \mid \Pi_{i\in I}\; k \rhd K_i\,)\big) \qquad \text{laws for restrictions in Table 2} \\ & \gtrsim & !\,k'(v).\big(\Pi_{k''\in K'}\,!\,\overline{k''}\langle v\rangle \mid \Pi_{k''\in\cup_{i\in I}K_i}!\overline{k''}\langle v\rangle)\big) \qquad \text{Lemma 6.5 (2)} \\ & \sim & P \end{array}$$

where in the last step, in the case when $K' \cap (\cup_{i\in I}K_i) \neq \emptyset$, Lemma 6.1 is needed. □

7 Operational Correspondences

This section is devoted to showing the close operational correspondence between the actions of a causal process A and those of the encoding standard process $[\![A]\!]$. In subsection 7.1 we study the operational correspondence on strong visible transitions (Proposition 7.7), and on strong silent transitions (Proposition 7.8). These results are used in subsection 7.2 to establish the correspondences on weak transitions.

7.1 Operational Correspondence on Strong Transitions

In the proof of the operational correspondence between A and $[\![A]\!]$ on strong transitions, we use Lemma 7.1 and Corollary 7.6. To prove the latter result, we rely on Lemmata 7.2 through 7.5. Lemma 7.1 and Lemma 7.2 are consequences of Lemma 6.4 and Lemma

6.6, dealing with compositions of wires. To see the relationship between Lemma 7.1 and Lemma 6.4, and between Lemma 7.2 and Lemma 6.6, remember that the parameter (K_0, k) is used in $[\![A]\!](K_0, k)$ to define wires of the form $k \rhd K'$, for some K'. Lemmata 7.1 and 7.2 are also useful when reasoning about processes returned by the encoding, in that they avoid us having to 'unfold' the definition of wire processes. The proofs of Lemmata 7.1 and 7.2 are reported in Appendix A.

Lemma 7.1 $[\![A]\!](K_0, k) \mid k \rhd K \gtrsim [\![A]\!](k, K_0, K) \mid k \rhd K$ $\square$

Lemma 7.2 *If* $k \notin K_0 \cup \big(\cup_{i \in I} K_i \big)$, *then*

$$\boldsymbol{\nu}\, k \left([\![A]\!](K_0, k) \mid \Pi_{i \in I}\, k \rhd K_i \right) \gtrsim [\![A]\!](K_0, \cup_i K_i) \qquad \square$$

The next three lemmata are used to prove Corollary 7.6. We have decomposed the corollary in this way to help understanding its meaning. (The corollary can also be proved directly, using Lemma 7.2 and transition induction; all cases of the induction would be fairly simple.) Lemma 7.3 is a variation of Lemma 7.2. The intuition of Lemma 7.4 is the following: Any cause in $[\![A]\!]K_0$ depends on K_0 (in the sense that there is a wire-connection from this cause to the causes in K_0); therefore, in the process $\boldsymbol{\nu}\, k \left([\![A]\!]K_0 \mid \Pi_{i \in I}\, k \rhd (K_i, K_0) \right)$, the dependency of k on K_0, expressed in the wires $k \rhd (K_0, K_i)$, is superfluous. In Lemma 7.5, the transition $A \xrightarrow[K;k]{\mu} A'$ shows that k depends on K in A': Therefore, the wire $k \rhd K$ can be safely removed. The first two lemmata are easily proved by exhibiting appropriate expansion relations, in the same style as Lemma 7.2. Lemma 7.5 is proved by transition induction on $A \xrightarrow[K;k]{\mu} A'$.

Lemma 7.3 *If* $k \notin K \cup K_0$, *then* $\boldsymbol{\nu}\, k \left([\![A]\!]K_0 \mid k \rhd K \right) \gtrsim [\![A[k \rightsquigarrow K]]\!]K_0$. $\square$

Lemma 7.4 *If* $k \notin (\cup_{i \in I} K_i) \cup K_0$, *then*

$$\boldsymbol{\nu}\, k \left([\![A]\!]K_0 \mid \Pi_{i \in I}\, k \rhd (K_i, K_0) \right) \gtrsim \boldsymbol{\nu}\, k \left([\![A]\!]K_0 \mid \Pi_{i \in I}\, k \rhd K_i \right). \qquad \square$$

Lemma 7.5 *If* $A \xrightarrow[K;k]{\mu} A'$ *with* $k \notin \mathcal{K}(A)$, *and* $k \notin K_0 \cup K'$, *then*

$$\boldsymbol{\nu}\, k \left([\![A']\!]K_0 \mid k \rhd K \mid k \rhd K' \right) \gtrsim \boldsymbol{\nu}\, k \left([\![A']\!]K_0 \mid k \rhd K' \right). \qquad \square$$

Corollary 7.6 *If* $A \xrightarrow[K;k]{\mu} A'$ *with* $k \notin \mathcal{K}(A)$, *and* $k \notin K_0 \cup K'$, *then*

$$\boldsymbol{\nu}\, k \left([\![A']\!]K_0 \mid k \rhd (K_0, K) \mid k \rhd (K_0, K') \right) \gtrsim [\![A'[k \rightsquigarrow K']]\!]K_0 .$$

PROOF: Using Lemmata 7.4, 7.5 and 7.3, in the order. □

Propositions 7.7 and 7.8 below are slightly stronger than we shall need. They relate the transitions of a causal process A with those of a standard process $[\![A]\!]K$, for a generic parameter K, while we shall only need the results for $K = \emptyset$. These more general statements are easier to handle in the inductive proofs (especially when dealing with rule CAU).

Proposition 7.7 (correspondence on strong visible transitions)

1. *(a) If* $A \xrightarrow[H;k]{a\langle\widetilde{b}\rangle} A'$ *then* $[\![A]\!]K \xrightarrow{a\langle\widetilde{bk}\rangle} \gtrsim k \rhd (H,K) \mid [\![A']\!]K$.

   (b) If $k \notin \mathcal{K}(A) \cup K$ *and* $A \xrightarrow[H;k]{\boldsymbol{\nu}\,\widetilde{b'}\,\overline{a}\langle\widetilde{b}\rangle} A'$ *then* $[\![A]\!]K \xrightarrow{(\boldsymbol{\nu}\,\widetilde{b'}k)\overline{a}\langle\widetilde{bk}\rangle} \gtrsim k \rhd (H,K) \mid [\![A']\!]K$.

2. *The converse of (1), on the actions from* $[\![A]\!]K$*:*

   (a) If $[\![A]\!]K \xrightarrow{a\langle\widetilde{bk}\rangle} P$ *then there exist* A', H *s.t.* $A \xrightarrow[H;k]{a\langle\widetilde{b}\rangle} A'$ *and* $P \gtrsim k \rhd (H,K) \mid [\![A']\!]K$.

   (b) If $[\![A]\!]K \xrightarrow{(\boldsymbol{\nu}\,\widetilde{b'}k)\overline{a}\langle\widetilde{bk}\rangle} P$ *then there exist* A', H *s.t.* $A \xrightarrow[H;k]{\boldsymbol{\nu}\,\widetilde{b'}\,\overline{a}\langle\widetilde{b}\rangle} A'$ *and* $P \gtrsim k \rhd (H,K) \mid [\![A']\!]K$.

PROOF: The proof exploits Lemma 7.1. We only deal with case (1.a), since the others are similar or easier. We proceed by induction on the derivation of $A \xrightarrow[H;k]{a\langle\widetilde{b}\rangle} A'$.

Case 1: INP rule.

(This is the base case.) Then we have $A = a(\widetilde{c}).\,P \xrightarrow[\emptyset\,;k]{a\langle\widetilde{b}\rangle} k :: P\{\widetilde{b}/\widetilde{c}\}$. Therefore

$$\begin{array}{lclcl} [\![A]\!]K & = & a(\widetilde{c}k).\,(\,k \rhd K \mid [\![P]\!]\{k\}) & \xrightarrow{a\langle\widetilde{bk}\rangle} & k \rhd K \mid [\![P\{\widetilde{b}/\widetilde{c}\}]\!]\{k\} \\ & \gtrsim & k \rhd K \mid [\![P\{\widetilde{b}/\widetilde{c}\}]\!](k,K) & = & k \rhd K \mid [\![k :: P\{\widetilde{b}/\widetilde{c}\}]\!]K \end{array}$$

where the use of $\gtrsim$ is due to Lemma 7.1.

Case 2: PAR rule.

Then $A = A_1 \mid A_2$ and, supposing that the move originates from A_1, the last rule applied is of the form

$$\text{PAR}: \frac{A_1 \xrightarrow[H;k]{a\langle\widetilde{b}\rangle} A_1'}{A_1 \mid A_2 \xrightarrow[H;k]{a\langle\widetilde{b}\rangle} A_1' \mid A_2}.$$

From the inductive hypothesis, we know that for all K there is P s.t.

$$\llbracket A_1 \rrbracket K \xrightarrow{a\langle\widetilde{bk}\rangle} P \gtrsim k \rhd (H,K) \mid \llbracket A'_1 \rrbracket K\,.$$

From this, using rule `S-PAR`, we get

$$\llbracket A_1 \mid A_2 \rrbracket K \xrightarrow{a\langle\widetilde{bk}\rangle} P \mid \llbracket A_2 \rrbracket K \gtrsim k \rhd (H,K) \mid \llbracket A'_1 \mid A_2 \rrbracket K$$

which is the thesis.

Case 3: `CAU` rule.

Then $A = K_0 :: A'$. The last rule applied in deriving the transition from A is of the form

$$\texttt{CAU}: \frac{A' \xrightarrow[H';k]{a\langle\widetilde{b}\rangle} A''}{A = K_0 :: A' \xrightarrow[H;k]{a\langle\widetilde{b}\rangle} K_0 :: A''}$$

where $H = K_0 \cup H'$. From the inductive hypothesis, for each K' there exists some P such that

$$\llbracket A' \rrbracket K' \xrightarrow{a\langle\widetilde{bk}\rangle} P \gtrsim k \rhd (H',K') \mid \llbracket A'' \rrbracket K'\,.$$

Recalling that $\llbracket A \rrbracket K = \llbracket A' \rrbracket (K_0, K)$ and letting $K' = K_0 \cup K$ in the above transition, the thesis follows.

Case 4: `OUT`, `SUM`, `RES`, `OPEN` or `REPL`.

These cases easily follow from the inductive assumption. □

Proposition 7.8 (correspondence on strong silent transitions)

1. *If* $A \xrightarrow{\tau} A'$, *then* $\llbracket A \rrbracket K \xrightarrow{\tau} \gtrsim \llbracket A' \rrbracket K$.

2. *If* $\llbracket A \rrbracket K \xrightarrow{\tau} P$, *then there exists* A' *such that* $A \xrightarrow{\tau} A'$ *and* $P \gtrsim \llbracket A' \rrbracket K$.

PROOF: We only deal with (1), since (2) is similar. The proof is an induction on the derivation of $A \xrightarrow{\tau} A'$ and exploits the previous proposition (7.7), Lemma 6.2 and Corollary 7.6. The most difficult case is when `COM` is the last rule applied:

$$\texttt{COM}: \frac{A_1 \xrightarrow[H_1;k]{a\langle\widetilde{b}\rangle} A'_1, \qquad A_2 \xrightarrow[H_2;k]{(\boldsymbol{\nu}\,\widetilde{c})\overline{a}\langle\widetilde{b}\rangle} A'_2}{A = A_1 \mid A_2 \xrightarrow{\tau} \boldsymbol{\nu}\,\widetilde{c}(A'_1[k \leadsto H_2] \mid A'_2[k \leadsto H_1]) \overset{def}{=} A'} \tag{4}$$

where $\widetilde{c} \subseteq \widetilde{b}$ and $k \notin \mathcal{K}(A)$. By applying Lemma 4.5 to rename k, if needed, we can also suppose $k \notin K$. From Proposition 7.7(1), applied to the premises of (4) we get

$$\begin{array}{llll} \llbracket A_1 \rrbracket K & \xrightarrow{a\langle \widetilde{b}k\rangle} \gtrsim & k \rhd (H_1, K) \mid \llbracket A'_1 \rrbracket K \overset{def}{=} R_1 \\ \llbracket A_2 \rrbracket K & \xrightarrow{(\boldsymbol{\nu}\, ck)\overline{a}\langle \widetilde{b}k\rangle} \gtrsim & k \rhd (H_2, K) \mid \llbracket A'_2 \rrbracket K \overset{def}{=} R_2 \,. \end{array}$$

From the above two transitions and rule `S-COM` we infer

$$\begin{array}{lll} \llbracket A \rrbracket K = \llbracket A_1 \rrbracket K \mid \llbracket A_2 \rrbracket K \xrightarrow{\tau} & \gtrsim & (\boldsymbol{\nu}\, \widetilde{c}k)(R_1 \mid R_2) \\ & \sim & (\boldsymbol{\nu}\, \widetilde{c}k)(\ k \rhd (H_1, K) \ \mid\ k \rhd (H_2, K) \ \mid \llbracket A'_1 \rrbracket K \mid \llbracket A'_2 \rrbracket K) \\ & \overset{def}{=} & P \,. \end{array}$$

Now, k can occur free in $\llbracket A'_i \rrbracket K$ ($i \in \{1, 2\}$) only in output subject position; thus we can apply Lemma 6.2 (1) to distribute the restricted replications $k \rhd (H_1, K) \mid k \rhd (H_2, K)$ over the parallel composition $\llbracket A'_1 \rrbracket K \mid \llbracket A'_2 \rrbracket K$ in P, obtaining:

$$\begin{array}{lll} P & \sim & \boldsymbol{\nu}\, \widetilde{c}\, \Pi_{i=1,2} \big(\boldsymbol{\nu}\, k\, (\ k \rhd (H_1, K) \ \mid\ k \rhd (H_2, K) \ \mid \llbracket A'_i \rrbracket K) \big) \\ & \overset{def}{=} & \boldsymbol{\nu}\, \widetilde{c}\, (S_1 \mid S_2) \,. \end{array}$$

Now, since $A_i \ \underset{H_i;\, k}{\xrightarrow{\mu}}\ A'_i$ and $k \notin K \cup H_i \cup \mathcal{K}(A_i)$, ($i \in \{1, 2\}$), we can apply Corollary 7.6 to S_i, so to replace k with H_j inside $\llbracket A'_i \rrbracket$ and eliminate the wires, thus:

$$S_i \gtrsim \llbracket A'_i [k \rightsquigarrow H_j] \rrbracket K$$

where if $i = 1$ then $j = 2$, and if $i = 2$ then $j = 1$. Since parallel composition and restriction preserve $\gtrsim$, $\boldsymbol{\nu}\, \widetilde{c}\, (S_1 \mid S_2) \gtrsim \boldsymbol{\nu}\, \widetilde{c}\, (\llbracket A'_1 [k \rightsquigarrow H_2] \rrbracket K \mid \llbracket A'_2 [k \rightsquigarrow H_1] \rrbracket K) = \llbracket A' \rrbracket K$ olds. Summarising, we have obtained that $\llbracket A \rrbracket K \xrightarrow{\tau} \ \gtrsim P \gtrsim \llbracket A' \rrbracket K$. This concludes the case. $\square$

7.2 Operational Correspondences on Weak Transitions

In this subsection we study the operational correspondence between encoded and encoding processes on weak transitions.

Proposition 7.9 (correspondence on weak silent transitions)

1. *If* $A \Longrightarrow A'$, *then* $\llbracket A \rrbracket \Longrightarrow \ \gtrsim \llbracket A' \rrbracket$.

2. *If* $\llbracket A \rrbracket \Longrightarrow P$, *then there exists* A' *such that* $A \Longrightarrow A'$ *and* $P \gtrsim \llbracket A' \rrbracket$.

PROOF:

1. It must be $A \xrightarrow{\tau^n} A'$, for some $n \geq 0$. We proceed by induction on n. For $n = 0$ the statement holds trivially. Suppose $n > 0$; then, for some A'', we have $A \xrightarrow{\tau^{n-1}} A'' \xrightarrow{\tau} A'$. From the inductive hypothesis, for some Q,

   $$[\![A]\!] \Longrightarrow Q \gtrsim [\![A'']\!] .$$

   Now, from Proposition 7.8(1) and $A'' \xrightarrow{\tau} A'$ we have that, for some P,

   $$[\![A'']\!] \xrightarrow{\tau} P \gtrsim [\![A']\!] .$$

   Since $Q \gtrsim [\![A'']\!]$, from the above transition we get, for some Q',

   $$Q \overset{\tau}{\Longrightarrow} Q' \gtrsim P .$$

   From this and $P \gtrsim [\![A']\!]$, we get $Q' \gtrsim [\![A']\!]$. To sum up, we have obtained:

   $$[\![A]\!] \Longrightarrow Q \overset{\tau}{\Longrightarrow} Q' \gtrsim [\![A']\!]$$

   which proves the thesis.

2. For some $n \geq 0$ it holds that $[\![A]\!] \xrightarrow{\tau^n} P$. Again, we proceed by induction on n. The case $n = 0$ is trivial. If $n > 0$, the for some Q, we have $[\![A]\!] \xrightarrow{\tau^{n-1}} Q \xrightarrow{\tau} P$. By the inductive hypothesis, we have, for some A'',

   $$A \Longrightarrow A'' \quad \text{with } Q \gtrsim [\![A'']\!] .$$

   From this and $Q \xrightarrow{\tau} P$, we deduce that, for some R,

   $$[\![A'']\!] \xrightarrow{\hat{\tau}} R \quad \text{with } P \gtrsim R .$$

   Now, by Proposition 7.8(2), there is A' s.t.

   $$A'' \xrightarrow{\hat{\tau}} A' \quad \text{with } R \gtrsim [\![A']\!].$$

   Thus, we have found A' s.t. $A \Longrightarrow A'$ and $P \gtrsim [\![A']\!]$, and proved the thesis. □

Remark 7.10 In the proof of item (2) of the above proposition the use of the expansion relation turns out to be necessary to close up the induction of the proof. Had we used $\approx$ in place of $\gtrsim$ in the above proof, from $Q \approx [\![A'']\!]$ and $Q \xrightarrow{\tau} P$, we could have only inferred $[\![A'']\!] \Longrightarrow R$ (in place of the stronger $[\![A'']\!] \xrightarrow{\hat{\tau}} R$); as a consequence, we could not have applied Proposition 7.8 to close up the induction. □

Proposition 7.11 (correspondence on weak visible transition)

1. (a) *If* $A \overset{a\langle\widetilde{b}\rangle}{\underset{K;k}{\Longrightarrow}} A'$, *then* $\llbracket A\rrbracket \overset{a\langle\widetilde{bk}\rangle}{\Longrightarrow} \;\gtrsim\; k \triangleright K \mid \llbracket A'\rrbracket$.

   (b) *If* $k \notin \mathcal{K}(A)$ *and* $A \overset{(\boldsymbol{\nu}\,\widetilde{b'})\overline{a}\langle\widetilde{b}\rangle}{\underset{K;k}{\Longrightarrow}} A'$, *then* $\llbracket A\rrbracket \overset{(\boldsymbol{\nu}\,\widetilde{b'}k)\overline{a}\langle\widetilde{bk}\rangle}{\Longrightarrow} \;\gtrsim\; k \triangleright K \mid \llbracket A'\rrbracket$.

2. *The converse of (1), on the actions from* $\llbracket A\rrbracket$*:*

   (a) *If* $\llbracket A\rrbracket \overset{a\langle\widetilde{bk}\rangle}{\Longrightarrow} P$, *then there exist* A' *and* K *s.t.* $A \overset{a\langle\widetilde{b}\rangle}{\underset{K;k}{\Longrightarrow}} A'$ *and* $P \gtrsim\; k \triangleright K \mid \llbracket A'\rrbracket$.

   (b) *If* $\llbracket A\rrbracket \overset{(\boldsymbol{\nu}\,\widetilde{b'}k)\overline{a}\langle\widetilde{bk}\rangle}{\Longrightarrow} P$, *then there exist* A' *and* K *s.t.* $A \overset{\boldsymbol{\nu}\,\widetilde{b'}\,\overline{a}\langle\widetilde{b}\rangle}{\underset{K;k}{\Longrightarrow}} A'$ *and* $P \gtrsim\; k \triangleright K \mid \llbracket A'\rrbracket$.

PROOF: We only consider case (1.a); the remaining ones are similar. For some A'' and A''' we have

$$A \Longrightarrow A'' \overset{a\langle\widetilde{b}\rangle}{\underset{K;k}{\longrightarrow}} A''' \Longrightarrow A'.$$

From Proposition 7.9, we have, for some P'',

$$\llbracket A\rrbracket \Longrightarrow P'' \gtrsim \llbracket A''\rrbracket .$$

Furthermore, from Proposition 7.7, for some Q,

$$\llbracket A''\rrbracket \overset{a\langle\widetilde{bk}\rangle}{\longrightarrow} Q \gtrsim\; k \triangleright K \mid \llbracket A'''\rrbracket .$$

Therefore, since $P'' \gtrsim \llbracket A''\rrbracket$, there exists P''' such that:

$$P'' \overset{a\langle\widetilde{bk}\rangle}{\Longrightarrow} P''' \gtrsim Q .$$

Since $A''' \Longrightarrow A'$, by Proposition 7.9(1), $\llbracket A'''\rrbracket \Longrightarrow\; \gtrsim \llbracket A'\rrbracket$. Then, using rule `S-PAR`, we get, for some Q',

$$k \triangleright K \mid \llbracket A'''\rrbracket \Longrightarrow Q' \gtrsim\; k \triangleright K \mid \llbracket A'\rrbracket \tag{5}$$

We have proved that $P''' \gtrsim Q \gtrsim\; k \triangleright K \mid \llbracket A'''\rrbracket$. From this and (5) we deduce that there is P' s.t.

$$P''' \Longrightarrow P' \gtrsim Q' \gtrsim\; k \triangleright K \mid \llbracket A'\rrbracket .$$

To sum up, we have

$$\llbracket A\rrbracket \Longrightarrow P'' \overset{a\langle\widetilde{bk}\rangle}{\Longrightarrow} P''' \Longrightarrow P' \gtrsim\; k \triangleright K \mid \llbracket A'\rrbracket$$

which concludes the case.

□

Remark 7.12

1. Suppose $[\![A]\!] \Longrightarrow P$; then P cannot perform actions along a cause name k. This is a consequence of Proposition 7.9 (2) and of the fact that the processes returned by the encoding cannot immediately perform actions along a cause name (the latter fact holds by the definition of the encoding).
2. Suppose $[\![A]\!] \overset{\mu}{\Longrightarrow} P$; then P cannot perform *output* actions along a cause name k. This is a consequence of Proposition 7.11(2) and of the item (1) of this remark. $\Box$

8 Full Abstraction

In the first part of the section we prove two results about wires processes, the *Cancelation Lemma* and the *Saturation Lemma*, that will be crucial in the second part to prove the full abstraction theorem.

8.1 The Cancelation and the Saturation Lemmata

The main result of this section is the Cancelation Lemma. It is very useful when verifying observation equivalence between processes returned by the encoding, to cancel wires which appear at the outermost level. The Cancelation Lemma will be used in combination with the *Saturation Lemma* in the proof of soundness for the encoding.

To state the two lemmata, we need two auxiliary notions: *Cause-Sets* of a process and *Saturation*. The cause-sets of a process A contains all sets of causes which can appear in transitions from A; that is, if $A \xrightarrow[K;k]{\mu} A'$, then K is in the cause-sets of A (this property is formalised in Lemma B.3 (1), in Appendix B).

Definition 8.1 (Cause-Sets of a process) *For a causal process A,* the cause-sets of *A, written $\mathcal{CS}(A)$, is the set of sets of causes inductively defined as follows:*

$$\begin{array}{rcl}
\mathcal{CS}(P) & \overset{def}{=} & \{\emptyset\} \\
\mathcal{CS}(K :: A) & \overset{def}{=} & \{K \cup H \,|\, H \in \mathcal{CS}(A)\} \\
\mathcal{CS}(\boldsymbol{\nu}\, a\, A) & \overset{def}{=} & \mathcal{CS}(A) \\
\mathcal{CS}(A_1 \mid A_2) & \overset{def}{=} & \mathcal{CS}(A_1) \cup \mathcal{CS}(A_2)\,.
\end{array}$$

Note that in general not all sets in $\mathcal{CS}(A)$ need to appear in causal transition from A; e.g., if A is $\boldsymbol{\nu}\, a\,(K :: a.\mathbf{0})$, then $K \in \mathcal{CS}(A)$, but A cannot perform any transition.

Now, the definition of saturation. Intuitively, a causal process A is saturated for a product of wires W if each set of causes H which appear in A is "closed" w.r.t. the causality relation encoded by W; i.e., if a wire in W has its entrance-point in H, then all end-points of the wire are in H.

Definition 8.2 (Saturation) *Let A be a causal process and let $W \overset{def}{=} \Pi_{i\in I}\ k_i \rhd K_i$. We say that A is* saturated *for W if for each $H \in \mathcal{CS}(A)$ and $i \in I$, $k_i \in H$ implies $K_i \subseteq H$.*

We can now state the Saturation and the Cancelation lemmata; their proofs are reported in Appendix B.

Lemma 8.3 (Saturation Lemma) *Let A be a causal process saturated for the product of wires $W \overset{def}{=} \Pi_{i\in I}\ k_i \rhd K_i$.*

1. *If $A \xrightarrow[K;k]{\mu} A'$ with k fresh, then A' is saturated for $k \rhd K \mid W$.*
2. *If $A \xrightarrow{\tau} A'$, then A' is saturated for W.*
3. *If $A \overset{\mu}{\underset{K;k}{\Longrightarrow}} A'$ with k fresh, then A' is saturated for $k \rhd K \mid W$.* □

Lemma 8.4 (Cancelation Lemma) *Let A be saturated for $k \rhd K$ and B be saturated for $k \rhd K'$. Then*

$$k \rhd K \mid [\![A]\!] \approx k \rhd K' \mid [\![B]\!] \ \text{ implies } \ K = K' \ \text{ and } \ [\![A]\!] \approx [\![B]\!] . \qquad \square$$

8.2 Soundness and Completeness of the encoding

Theorem 8.5 (Soundness) $[\![A]\!] \approx [\![B]\!]$ *implies $A \approx_c B$.*

Proof: We prove that the relation

$$\mathcal{R} = \{(A, B) \ : \ [\![A]\!] \approx [\![B]\!]\}$$

is a $\approx_c$-bisimulation. Let $(A, B) \in \mathcal{R}$. We only show how the moves of A are matched by B, since the relation is symmetrical. We check clauses (1) and (2) of the definition of $\approx_c$.

Clause (1)

$A \Longrightarrow A'$ implies, from Proposition 7.9(1)

$\llbracket A \rrbracket \Longrightarrow P$, for some P with $P \approx \llbracket A' \rrbracket$. Since $\llbracket A \rrbracket \approx \llbracket B \rrbracket$

$\llbracket B \rrbracket \Longrightarrow Q$, for some Q with $P \approx Q$. From Proposition 7.9(2) we get

$B \Longrightarrow B'$, for some B' with $Q \approx \llbracket B' \rrbracket$.

To sum up, we have $\llbracket A' \rrbracket \approx P \approx Q \approx \llbracket B' \rrbracket$. Hence $(A', B') \in \mathcal{R}$.

Clause (2) We only look at the input case; the output case is similar. We suppose that $k \notin \mathcal{K}(A, B)$.

$A \overset{a\langle \widetilde{b} \rangle}{\underset{K;k}{\Longrightarrow}} A'$ implies, from Proposition 7.11(1),

$\llbracket A \rrbracket \overset{a\langle \widetilde{bk} \rangle}{\Longrightarrow} P$, for some P with $P \approx k \rhd K \mid \llbracket A' \rrbracket$. Since $\llbracket A \rrbracket \approx \llbracket B \rrbracket$, we get

$\llbracket B \rrbracket \overset{a\langle \widetilde{bk} \rangle}{\Longrightarrow} Q$, for some Q with $Q \approx P$. From Proposition 7.11(2), we get

$B \overset{a\langle \widetilde{b} \rangle}{\underset{K';k}{\Longrightarrow}} B'$, for some Q with $Q \approx k \rhd K' \mid \llbracket B' \rrbracket$.

We have obtained that $k \rhd K \mid \llbracket A' \rrbracket \approx k \rhd K' \mid \llbracket B' \rrbracket$. From the Saturation Lemma applied to above two weak causal transitions, it follows that A' is saturated for $k \rhd K$ and that B' is saturated for $k \rhd K'$. From the Cancelation Lemma, it follows that $K = K'$ and $\llbracket A' \rrbracket \approx \llbracket B' \rrbracket$. Hence $(A', B') \in \mathcal{R}$, and we are done. $\square$

Theorem 8.6 (Completeness) *$A \approx_{\mathrm{c}} B$ implies $\llbracket A \rrbracket \approx \llbracket B \rrbracket$.*

Proof: We exploit the characterisation of causal bisimulation in terms of loose causal bisimulation ($\approx'_{\mathrm{c}}$) in Proposition 4.12 (we need $\approx'_{\mathrm{c}}$ for the treatment of clause (2) below). We prove that the relation

$$\mathcal{R} = \{(\llbracket A \rrbracket, \llbracket B \rrbracket) \; : \; A \approx'_{\mathrm{c}} B\}$$

is a weak bisimulation up to $\gtrsim$ and up to context. Let $(\llbracket A \rrbracket, \llbracket B \rrbracket) \in \mathcal{R}$. We look at clause (1) and (2) of the definition of observation equivalence, on the moves by $\llbracket A \rrbracket$.

Clause (1)

$\llbracket A \rrbracket \Longrightarrow P$ implies, from Proposition 7.11(2)

$A \Longrightarrow A'$, for some A' with $P \gtrsim \llbracket A' \rrbracket$. Since $A \approx'_{\mathrm{c}} B$

$B \Longrightarrow B'$, for some B' with $A' \approx'_{\mathrm{c}} B'$. From Proposition 7.11(1)

$\llbracket B \rrbracket \Longrightarrow Q \gtrsim \llbracket B' \rrbracket$, for some Q.

To sum up, we have $[\![A]\!] \Longrightarrow P \gtrsim [\![A']\!]$, $[\![B]\!] \Longrightarrow Q \gtrsim [\![B']\!]$, and $A' \approx'_{\mathrm{c}} B'$. Hence $([\![A']\!], [\![B']\!]) \in \mathcal{R}$, and we are done.

Clause (2) We check only the input case (the output case is similar, only note that in that case, one can suppose w.l.o.g. $k \notin \mathcal{K}(A, B)$):

$$\begin{array}{ll}
[\![A]\!] \overset{a\langle\widetilde{b}k\rangle}{\Longrightarrow} P & \text{implies, from Proposition 7.11(2),} \\
A \underset{K;k}{\overset{a\langle\widetilde{b}\rangle}{\Longrightarrow}} A', & \text{for some } A' \text{ with } P \gtrsim k \rhd K \mid [\![A']\!]. \text{ Since } A \approx'_{\mathrm{c}} B \\
B \underset{K;k}{\overset{a\langle\widetilde{b}\rangle}{\Longrightarrow}} B', & \text{for some } B' \text{ with } A' \approx'_{\mathrm{c}} B'. \text{ From Proposition 7.11(1)} \\
[\![B]\!] \overset{a\langle\widetilde{b}k\rangle}{\Longrightarrow} Q & \text{for some } Q' \text{ with } Q \gtrsim k \rhd K \mid [\![B']\!].
\end{array}$$

Processes $k \rhd K \mid [\![A']\!]$ and $k \rhd K \mid [\![B']\!]$ allow us to close the bisimulation $\mathcal{R}$, up to $\gtrsim$ and up to the static context $k \rhd K \mid [\,.\,]$. □

Theorems 8.5 and 8.6 prove the full abstraction of the encoding. That is, (using also Proposition 5.2) causal bisimulation on causal terms in $\mathcal{P}_{\mathrm{c}}(\mathcal{N}, \mathcal{K})$ and which respect a sorting Ob can be reconducted to the ordinary observation equivalence among standard terms in $\mathcal{P}(\mathcal{N}^+)$ and which respect a sorting $[\![Ob]\!]$, where $\mathcal{N}^+$ is an appropriate extension of the class of names $\mathcal{N}$.

In particular, this result holds for the standard processes in $\mathcal{P}_{\mathrm{c}}(\mathcal{N}, \mathcal{K})$, i.e., those terms in $\mathcal{P}_{\mathrm{c}}(\mathcal{N}, \mathcal{K})$ which do not contain the causal prefix $K :: A$. Formally, these are the processes in the class $\mathcal{P}(\mathcal{N})$. The applicability of the encoding to standard processes was the result we were most interested in.

Corollary 8.7 (Full abstraction of the encoding on standard processes) *For all* P *and* $Q \in \mathcal{P}(\mathcal{N})$, *we have* $[\![P]\!], [\![Q]\!] \in \mathcal{P}(\mathcal{N}^+)$ *and* $P \approx_{\mathrm{c}} Q$ *iff* $[\![P]\!] \approx [\![Q]\!]$. □

9 Examples of applications

In this section we give some examples of how the encoding and the full abstraction theorems can be used to reason about causal bisimulation.

Some simple laws for causal bisimulation follow directly from the definition of the encoding; for example:

$$\begin{aligned} K :: (A \mid B) &\approx_{\mathrm{c}} (K :: A) \mid (K :: B)\,, \\ K :: H :: A &\approx_{\mathrm{c}} K \cup H :: A\,, \\ \emptyset :: A &\approx_{\mathrm{c}} A\,. \end{aligned}$$

As an easy corollary of the definition of our encoding and its full abstraction, we can derive the congruence of $\approx_{\mathrm{c}}$ w.r.t. the causal prefix $K :: -$ and all operators which preserve $\approx$. For instance, the following sequence of implications, that holds for any causal processes A, B and C, provides the argument for parallel composition:

$$\begin{array}{lll} A \approx_{\mathrm{c}} B & \text{implies} & [\![A]\!] \approx [\![B]\!] \\ & \text{which implies} & [\![A \mid C]\!] = [\![A]\!] \mid [\![C]\!] \approx [\![B]\!] \mid [\![C]\!] = [\![B \mid C]\!]\,, \\ & \text{which implies} & A \mid C \approx_{\mathrm{c}} B \mid C\,. \end{array}$$

Similarly, we can use the encoding and the property that $\approx$ is not preserved by the sum operator to check that also $\approx_{\mathrm{c}}$ is not preserved by sum. We have

$$[\![\tau.a]\!] = \tau.[\![a]\!] \approx [\![a]\!]$$

and

$$[\![\tau.a + b]\!] = \tau.[\![a]\!] + [\![b]\!] \not\approx [\![a]\!] + [\![b]\!] = [\![a + b]\!]\,.$$

Hence $\tau.a \approx_{\mathrm{c}} a$, but $\tau.a + b \not\approx_{\mathrm{c}} a + b$, which gives us a counterexample for the congruence of $\approx_{\mathrm{c}}$ w.r.t. sum. Thus, the only congruence property for $\approx_{\mathrm{c}}$ remained is the one for the input prefix operator $a(\tilde{b}).-$. The known counter-examples showing that $a(\tilde{b}).-$ does not preserve $\approx$ involve either the *match* operator $[a = b]$ — not considered here — or the expansion law. For example, $a \mid b \approx a.\overline{b} + \overline{b}.a$, but $c(a).(a \mid \overline{b}) \not\approx c(a).(a.\overline{b} + \overline{b}.a)$, since the process $c(a).(a \mid \overline{b})$, after receiving b, can perform a communication, whereas $c(a).(a.\overline{b} + \overline{b}.a)$ cannot. The same kind of counter-example does not apply to $\approx_{\mathrm{c}}$, as the expansion law $a \mid b \approx a.\overline{b} + \overline{b}.a$ does not hold for this equivalence. Indeed, the problem of input prefix for $\approx_{\mathrm{c}}$ appears non-trivial. A discussion on this topic is postponed to the concluding section.

More complex laws can be proved by taking advantage of the well-understood algebraic theory of $\approx$. Below, we show an example of this. Consider $P \overset{def}{=} \boldsymbol{\nu}\, b\,(a.b.c \mid \overline{b}.d)$ and $Q \overset{def}{=} \boldsymbol{\nu}\, b\,(a.b.d \mid \overline{b}.c)$. Processes P and Q differ only because c and d are placed at different locations. However, in both cases c and d have the same cause, namely a. Indeed, it holds that $P \approx_{\mathrm{c}} Q$. We can prove this equality via algebraic manipulations, by showing that $\llbracket P \rrbracket \approx \llbracket Q \rrbracket$. If

$$R \overset{def}{=} \; h \triangleright \emptyset \;\; \mid \llbracket d \rrbracket h \tag{6}$$

then we have

$$\llbracket P \rrbracket = \boldsymbol{\nu}\, b \Big(a(k).\, (\; k \triangleright \emptyset \;\; \mid \llbracket b.c \rrbracket k) \mid \boldsymbol{\nu}\, h\, \overline{b} \langle h \rangle .\, R \Big) . \tag{7}$$

From (7), applying the expansion law and simple laws for restriction (Table 2) we obtain

$$\llbracket P \rrbracket \sim a(k).\, \Big(\; k \triangleright \emptyset \;\; \mid \boldsymbol{\nu}\, b\, (\llbracket b.c \rrbracket k \mid \boldsymbol{\nu}\, h\, \overline{b} \langle h \rangle .\, R) \Big) . \tag{8}$$

Since, by definition,

$$\llbracket b.c \rrbracket k = b(h).\, (\; h \triangleright k \;\; \mid \llbracket c \rrbracket h) \tag{9}$$

applying expansion and restriction laws we obtain

$$\begin{array}{rcl} \boldsymbol{\nu}\, b\, (\llbracket b.c \rrbracket k \mid \boldsymbol{\nu}\, h\, \overline{b} \langle h \rangle .\, R) & \sim & \\ \tau . \boldsymbol{\nu}\, h \Big(\; h \triangleright k \;\; \mid \llbracket c \rrbracket h \mid R \Big) & = & \\ \tau . \boldsymbol{\nu}\, h \Big(\; h \triangleright k \;\; \mid \llbracket c \rrbracket h \mid \; h \triangleright \emptyset \;\; \mid \llbracket d \rrbracket h \Big) & \overset{def}{=} & T\, . \end{array} \tag{10}$$

Rearranging subterms, we can write

$$T \sim \tau . \boldsymbol{\nu}\, h \Big(\; h \triangleright k \;\; \mid \; h \triangleright \emptyset \;\; \mid \llbracket c \mid d \rrbracket h \Big) . \tag{11}$$

Applying Lemma 7.2 to the right-hand side of (11):

$$T \gtrsim \tau . \llbracket c \mid d \rrbracket k\, . \tag{12}$$

From (10) and (12), and since $\gtrsim$ is preserved by parallel composition,

$$k \triangleright \emptyset \;\; \mid \boldsymbol{\nu}\, b\, (\llbracket b.c \rrbracket k \mid \boldsymbol{\nu}\, h\, \overline{b} \langle h \rangle .\, R) \gtrsim \; k \triangleright \emptyset \;\; \mid \tau . \llbracket c \mid d \rrbracket k\, . \tag{13}$$

Relation (13) holds or any cause k. From this, due to the congruence property of $\gtrsim$, and from (8), we get:

$$\llbracket P \rrbracket \gtrsim a(k).\, \Big(\; k \triangleright \emptyset \;\; \mid \tau . \llbracket c \mid d \rrbracket k \Big) . \tag{14}$$

In a symmetric way — exchanging the roles of c and d — one proves that

$$\llbracket Q \rrbracket \gtrsim a(k).\, \Big(\; k \triangleright \emptyset \;\; \mid \tau . \llbracket d \mid c \rrbracket k \Big) . \tag{15}$$

But by the definition of encoding and by the commutativity law for parallel composition operator, we have

$$[\![c \mid d]\!]k \sim [\![d \mid c]\!]k \, ,$$

for any k. Hence,

$$a(k).\left(\, k \rhd \emptyset \quad \mid \tau.[\![c \mid d]\!]k\right) \sim a(k).\left(\, k \rhd \emptyset \quad \mid \tau.[\![d \mid c]\!]k\right). \qquad (16)$$

Finally, we deduce the original claim $[\![P]\!] \approx [\![Q]\!]$ from (14), (15) and (16).

The above sequence of manipulations consists of applications of familiar algebraic laws (expansion and restriction laws), plus an application of the absorption Lemma 7.2, to manipulate wire processes. In particular, Lemma 7.2 avoided us having to unfold the definition of wire processes and hence having explicitly to deal with the replication operator. A proof of $P \approx_{\mathrm{c}} Q$ is also possible utilising a proof system for $\approx_{\mathrm{c}}$ on CCS, like Kiehn's [Kie93]. However, the laws for the interleaving $\approx$ are simpler than those for the non-interleaving $\approx_{\mathrm{c}}$, and do not require the introduction of extra operators. Kiehn's system, besides the causal prefix $K :: -$, uses the *left merge* and the *communication merge* operators, to express an appropriate form of expansion law for causal bisimulation. Moreover, Kiehn's system is not purely equational — it includes non-trivial inference rules.

10 Comparing the encodings of location and causal bisimulation

We have presented an encoding of the non-interleaving causal bisimulation into the interleaving observation equivalence in the π-calculus. In a previous paper [San94b], the same issue has been tackled for *location bisimulation.* Causal bisimulation and location bisimulation aim, respectively, to capture the causal and the spatial dependencies on processes. Location bisimulation can be described using a syntax and an operational semantics similar to those for causal bisimulation in Definition 4.9 and Table 3. What distinguishes location bisimulation is that an action μ causes a later action μ' only if μ' emanates from a position (called *location*) syntactically underneath μ's position. In this framework, a communication between two subterms of a process has no causal effect whatsoever because the spatial disposition of the process components is left unchanged.

The processes

$$P_1 \overset{def}{=} a \mid \overline{c} \qquad\qquad \text{and} \qquad\qquad P_2 \overset{def}{=} \boldsymbol{\nu}\, b\,(a.\, b|\overline{b}.\, c) + \boldsymbol{\nu}\, b\,(c.\, b|\overline{b}.\, a)$$

are location bisimilar: Actions a and c are unrelated both in P_1 and P_2, because they emanate from unrelated locations. However, P_1 and P_2 are not causally bisimilar: Only in P_2 the firing of either a or c is necessary to liberate the other action. On the other hand, and for opposite reasons, processes

$$Q_1 \overset{def}{=} \boldsymbol{\nu}\, b\,(a.b.c|\overline{b}.d) \qquad\qquad \text{and} \qquad\qquad Q_2 \overset{def}{=} \boldsymbol{\nu}\, b\,(a.b.d|\overline{b}.c)$$

are causally bisimilar (see Section 9) but not location bisimilar. The contrasts between the two equivalences are only imputable to the treatment of communications between processes. Indeed, Aceto's work [Ace92b] shows that they coincide in a calculus with no such communications. Affinities and differences are evidenced in their encodings into observation equivalence presented here and in [San94b]. If we limit ourselves to standard processes, then the two encodings act as a homomorphism on all operators but input and output prefixes. There is a basic schema in the two encodings of the prefixes:

$$[\![a(\widetilde{b}).\, P]\!]k \quad \overset{def}{=} \qquad a(\widetilde{b}x).\Big(\ \boxed{> x, h}\ \ h \rhd k \ \ \mid [\![P]\!]h\Big) \tag{17}$$

$$[\![\overline{a}\langle\widetilde{b}\rangle.\, P]\!]k \quad \overset{def}{=} \quad (\boldsymbol{\nu}\, x)\overline{a}\langle\widetilde{b}x\rangle.\Big(\ \boxed{< x, h}\ \ h \rhd k \ \ \mid [\![P]\!]h\Big) \tag{18}$$

where h is a fresh name. Formally, $\boxed{> x, h}$ and $\boxed{< x, h}$ stand for unary operators on the argument $\ h \rhd k \ \ \mid [\![P]\!]h$. But it is more enlightening — and this explains the way in which they appear in (17) and (18) — to think of them as 'missing parts', still to be filled, of the expressions (17) and (18). Indeed, $\boxed{> x, h}$ and $\boxed{< x, h}$ play the role of the *access key* to the wire $\ h \rhd k\ $, which holds the causality information. In the case of location bisimulation, the access key is a restricted particle $\overline{x}\langle h\rangle$ in parallel with the wire, that is:

$$\text{“}\ \boxed{> x, h}\ \text{”} \quad \text{is} \quad \text{“}\boldsymbol{\nu}\, h\, \overline{x}\langle h\rangle \ \mid\text{”} \qquad \text{and} \tag{19}$$

$$\text{“}\ \boxed{< x, h}\ \text{”} \quad \text{is} \quad \text{“}\boldsymbol{\nu}\, h\, \overline{x}\langle h\rangle \ \mid\text{”}\,.$$

If the prefix $a(\widetilde{b}x)$ or $\overline{a}\langle\widetilde{b}x\rangle$ in (17-18) is consumed in a visible action, the external observer can use the key $\overline{x}\langle h\rangle$ to access name h and, with it, the wire $\ h \rhd k\ $. By contrast, if the two complementary prefixes $a(\widetilde{b}x)$ and $\overline{a}\langle\widetilde{b}x\rangle$ communicate with each other, the respective keys remain deadlocked, because of the same output polarity. Consequently, the related wires remain disconnected. This reflects the fact that in location bisimulation a communication between processes has no effect on causes. A few simplifications on this schema lead to the final encoding, reported in Table 10. (Note in particular the elimination of the innermost replication of the wire: This is possible because, in the case of location bisimulation, only 'linear' causal chains may be generated, hence at most one wire may get connected to the end-point of another wire.)

In the encoding of location bisimulation interactions between a process and the external observer and between processes are different — the key $\overline{x}\langle h\rangle$ is to be used only by the external observer. The encoding of causal bisimulation is obtained by eliminating this asymmetry. A key takes the same polarity as the related prefix. Writing "$\overline{x}(h).$" as abbreviation for the bound output "$\boldsymbol{\nu}\, h\, \overline{x}\langle h\rangle.$" to emphasise the symmetry, then

$$\begin{array}{lll} \text{“}\,\boxed{> x, h}\,\text{”} & \text{is} & \text{“}x(h).\text{”} \quad \text{and} \\ \text{“}\,\boxed{< x, h}\,\text{”} & \text{is} & \text{“}\overline{x}(h).\text{”}\,. \end{array} \tag{20}$$

Thus, if the prefix $a(\widetilde{b}x)$ or $\overline{a}\langle\widetilde{b}x\rangle$ in (17-18) produces a visible action, the key $x(h)$ or $\overline{x}(h)$ can be used by the external observer, in the same way as for location bisimulation, to access the wire $h \rhd k$. But now, if the prefixes $a(\widetilde{b}x)$ and $\overline{a}\langle\widetilde{b}x\rangle$ communicate with each other, the respective keys coalesce, because of opposite polarities. Consequently, the related wires get connected, which reflects the fact that in causal bisimulation a communication involves a merge of cause sets (the fact that a key is not blocking also justifies the sequentialisation between a key and the associated wire in (20), in place of the parallel composition as in (19)). In this schema of encoding, a key $x(h)$ or $\overline{x}\langle h\rangle$ only delays the access to the wire $h \rhd k$ one step. The encoding used in this paper is obtained by removing this level of indirection. To facilitate the comparison with the encoding location bisimulation, the clauses for input and output prefixes have been rewritten in Table 10.

11 Conclusions and future work

We have examined causality in the π-calculus and provided a fully abstract encoding which reconducts the non-interleaving causal bisimulation to the interleaving observation equivalence. This permits re-using all the easier mathematical theory of the latter to reason about the former. As a particular case, CCS causal bisimulation is reconducted to the observation equivalence of the monadic π-calculus.

Wire processes play a crucial role in modeling the pointer mechanism of causal bisimulation. The proofs of our main theorems rely heavily on the ability of manipulating wires, Encoding of Location Bisimulation; process $h \blacktriangleright k$ is $!\, h.\, \overline{k}$.

$$\llbracket a(\widetilde{b}).\, A \rrbracket k \stackrel{def}{=} a(\widetilde{b}x).\, \boldsymbol{\nu}\, h\, (\overline{x}\langle h\rangle \mid h \blacktriangleright k \mid \llbracket A \rrbracket h)$$

$$\llbracket \overline{a}\langle\widetilde{b}\rangle.\, A \rrbracket k \stackrel{def}{=} \boldsymbol{\nu}\, h\, \overline{a}\langle\widetilde{b}x\rangle.\, \boldsymbol{\nu}\, h\, (\overline{x}\langle h\rangle \mid h \blacktriangleright k \mid \llbracket A \rrbracket h)$$

……

Encoding of Causal Bisimulation.

$$\llbracket a(\widetilde{b}).\, A \rrbracket k \stackrel{def}{=} a(\widetilde{b}h).\, (\, h \triangleright k \mid \llbracket A \rrbracket h)$$

$$\llbracket \overline{a}\langle\widetilde{b}\rangle.\, A \rrbracket k \stackrel{def}{=} \boldsymbol{\nu}\, h\, \overline{a}\langle\widetilde{b}h\rangle.\, (\, h \triangleright k \mid \llbracket A \rrbracket h)$$

Table 10

as provided by the lemmata of Section 5. The proofs of these lemmata, in turn, rely on the "up-to techniques" of Section 3.

We believe that the full abstraction of our encoding extends to the congruence relations; i.e., if $\approx^{\mathrm{c}}$ and $\approx^{\mathrm{c}}_{\mathrm{c}}$ are the congruences induced by observation equivalence and causal bisimulation, respectively, then for all causal terms A and B, it holds that

$$A \approx^{\mathrm{c}}_{\mathrm{c}} B \quad \text{iff} \quad \llbracket A \rrbracket \approx^{\mathrm{c}} \llbracket B \rrbracket\, .$$

It should be possible to prove this exploiting the full abstraction on the bisimilarity relations. The result could then be used to examine whether a proof system for $\approx^{\mathrm{c}}_{\mathrm{c}}$ can be obtained from one for $\approx^{\mathrm{c}}$ plus, possibly, Lemmata 6.4, 6.6, 7.1 and 7.2 to manipulate wires.

An interesting issue is the study of the congruence properties for the input-prefix operator in causal-sensitive behavioural equivalences. Intuitively, a relation over π-calculus processes is preserved by input prefix if it is preserved by name substitutions. This property usually fails if the *matching* operator is present, as for the calculus in [MPW92]. A matching $[a = b]P$ represents an `if-then` construct with guard "$a = b$". Thus, if a and b are different names, one might want to regard $[a = b]a.\mathbf{0}$ and $\mathbf{0}$ as equivalent since both are deadlocked, but $([a = b]a.\mathbf{0})\{a/b\}$ and $\mathbf{0}\{a/b\}$ as distinct since only the latter is deadlocked. Indeed, with the matching operator observation equivalence, location bisimulation and causal bisimulation are all not preserved by substitutions. However, in many situations matching can be avoided or simulated. Hence one might wish to consider a matching-free calculus, as we did in the present paper. On such a language, the congruence for substitutions still fails for observation equivalence and location bisimulation [San94b], but we conjecture that it holds for causal bisimulation. The failure for the former equivalences is essentially due to the fact that both equivalences may relate processes with different degrees of parallelism (consider, for instance, the processes P_1 and P_2 in Section 10); as a consequence, when a substitution applied to such two processes identifies previously distinct channels, new communication possibilities may be created in one process, but not in the other. However, it can be expected that the situation for $\approx_c$ is better, since $\approx_c$ does respect parallelism. Preliminary results indicate that in the matching-free calculus $\approx_c$ *is* indeed preserved by substitutions and hence by input prefix. If confirmed, this would have significant consequences, both practical — proving causal bisimulation equalities would be easier — and more theoretical — the proliferation of different forms of bisimulation, like the *early, late* and *open* of interleaving and location bisimilarities [PS93, San94a, San94b], would be avoided in this causality framework.

We have individuated two forms of causal dependency between π-calculus actions, which we called subject and object dependencies. The former originate from prefix-nesting and are propagated through interactions within processes, the latter originate from the name-binding mechanisms. Our formulation of causal bisimulation distinguishes between processes which differ for the subject or for the object dependencies. Thus we discriminate between $P \overset{def}{=} \boldsymbol{\nu}\, b \left(\overline{a}\langle b\rangle.\, \mathbf{0} \mid \overline{b}\langle y\rangle.\, \mathbf{0}\right)$ and $Q \overset{def}{=} \boldsymbol{\nu}\, b\, (\overline{a}\langle b\rangle.\, \overline{b}\langle y\rangle.\, \mathbf{0})$, because in P there is both a subject and an object dependency between the actions at a and at b, whereas in Q there is only an object dependency. We would like to compare this notion of causal bisimulation with one in which subject and object dependencies are still taken into account but not separated, so to equate P and Q. One possibility is to carry out this analysis using our encoding or variations of it.

Acknowledgments. We are most grateful to Rocco De Nicola for comments and discussions on the topic of the paper. Michele Boreale has been partially supported by the HCM Network "EXPRESS". Davide Sangiorgi's research has been supported by the ESPRIT BRA project 6454 "CONFER".

References

[Ace92a] L. Aceto. History preserving, Causal and Mixed-Ordering Equivalence over Stable Event Structure (Note). in *Fundamentae Informaticae*, 17(4):319-331, 1992.

[Ace92b] L. Aceto. Relating Distributed, Temporal and Causal Observations of Simple Processes. in *Fundamentae Informaticae*, 17(4):369-397, 1992.

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

[BCHK91] G. Boudol, I. Castellani, M. Hennessy, and A. Kiehn. A theory of processes with localities. To appear in *Formal Aspects of Computing.*

[BD92] M. Boreale and R. De Nicola. Testing equivalence for mobile processes. In R. Cleaveland, editor, *3rd CONCUR*, volume 630 of *Lecture Notes in Computer Science*, pages 2–16. Springer Verlag, 1992. To appear in *Information and Computation.*

[BS94] M. Boreale and D. Sangiorgi. Full version of this paper. To appear as a Technical Report, Department of Computer Science, University of Edinburgh, 1994.

[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.

[DDM88] P. Degano , R. De Nicola and U. Montanari. Partial Ordering Descriptions and Observations of Concurrent Processes. in *Linear Time, Branching Time and Partial Order in Logics and Models of Concurrency*, volume 354 of *Lecture Notes in Computer Science*, pages 438–466. Springer-Verlag, 1988.

[DDM93] P. Degano , R. De Nicola and U. Montanari. Observation Trees. in *Proc. North American Conference on Process Algebras (NAPAW) '92*, Springer-Verlag, 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.

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

[Kie93] A. Kiehn. Proof systems for cause based equivalences. In *Proc. MFCS 93*, volume 711 of *Lecture Notes in Computer Science*. Springer Verlag, 1993.

[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.

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

[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.

[Par81] D.M. Park. Concurrency on automata and infinite sequences. In P. Deussen, editor, *Conf. on Theoretical Computer Science*, volume 104 of *Lecture Notes in Computer Science*. Springer Verlag, 1981.

[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.

[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. To appear in *Information and Compuation.*

[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.

[San93] D. Sangiorgi. A theory of bisimulation for the π-calculus. To appear in *Acta Informatica.* Extended Abstract in *Proc. CONCUR '93*, volume 715 of Lecture Notes in Computer Science, Springer Verlag.

[San94a] D. Sangiorgi. On the bisimulation proof method. Draft, 1994. LFCS, Edinburgh.

[San94b] D. Sangiorgi. Locality and non-interleaving semantics in calculi for mobile processes. Technical Report ECS–LFCS–94–282, LFCS, Dept. of Comp. Sci., Edinburgh Univ., 1994. An extract appeared in *Proc. TACS '94*, Lecture Notes in Computer Science, Springer Verlag.

[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.

[Wal94] D. Walker. Objects in the π-calculus. *Information and Computation*, 1994. To appear.

$$\begin{array}{llrcl}
\textbf{Abelian monoid laws for } +: & & P_1 + P_2 & \equiv & P_2 + P_1 \\
 & & P_1 + (P_2 + P_3) & \equiv & (P_1 + P_2) + P_3 \\
 & & P + \mathbf{0} & \equiv & P \\
\textbf{Abelian monoid laws for } \mid: & & P_1 \mid P_2 & \equiv & P_2 \mid P_1 \\
 & & P_1 \mid (P_2 \mid P_3) & \equiv & (P_1 \mid P_2) \mid P_3 \\
 & & P \mid \mathbf{0} & \equiv & P \\
\textbf{Restriction:} & & \boldsymbol{\nu}\, a\, \mathbf{0} & \equiv & \mathbf{0} \\
 & & \boldsymbol{\nu}\, a\, \boldsymbol{\nu}\, b\, P & \equiv & \boldsymbol{\nu}\, b\, \boldsymbol{\nu}\, a\, P \\
 & \text{if } a \notin fn(P_2), \text{ then} & (\boldsymbol{\nu}\, a\, P_1) \mid P_2 & \equiv & \boldsymbol{\nu}\, a\, (P_1 \mid P_2) \\
\textbf{Replication:} & & !\,P & \equiv & P \mid !\,P
\end{array}$$

Table 5: The rules of structural congruence ($\equiv$)

A Proofs of Lemmata 7.1 and 7.2.

To facilitate the proof of a few lemmata below, and to better capture the strength of their assertions we use the *structural congruence* relation. 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. Structural congruence [Mil91] is written $\equiv$ and is defined as the smallest congruence over processes which satisfies the rules in Table 5. These are self-evident rules which must be valid in any reasonable behavioural equivalence.

Lemma A.1 *The relation $\equiv$ is a strong bisimulation and is preserved by name substitution.* $\square$

A.1 Proofs of Lemma 7.1 and of Lemma 7.2

The following lemma allows us to rename names in the object part of input transitions.

Lemma A.2 (Renaming Lemma) *Suppose $P \xrightarrow{a\langle\tilde{b}\rangle} P'$. Then there are $\tilde{x}$ and P'' s.t. $P' = P''\{\tilde{b}/\tilde{x}\}$ and for all $\tilde{c}$, $P \xrightarrow{a\langle\tilde{c}\rangle} P''\{\tilde{c}/\tilde{x}\}$.*

Proof: Simple transition induction. □

Corollary A.3

1. *If* $P \xrightarrow{a\langle\widetilde{bh}\rangle} P'$, *and* $k \cap fn(P) = \emptyset$, *then also* $P \xrightarrow{a\langle\widetilde{bk}\rangle} P''$, *for some* P'' *with* $P''\{^h\!/_k\} = P'$.
2. *If* $P \xrightarrow{a\langle\widetilde{bk}\rangle} P''$ *and* $k \cap fn(P) = \emptyset$, *then, for all* h, *also* $P \xrightarrow{a\langle\widetilde{bh}\rangle} P' = P''\{^h\!/_k\}$. □

The proofs of Lemmata 7.1 and 7.2 use Lemmata 6.4 and 6.6. We also rely on Lemma A.5 below. This lemma, basically, says that the actions a process $[\![A]\!]K$ can perform do not depend from the parameter K. That is, two processes $[\![A]\!]K$ and $[\![A]\!]K'$ can perform the same actions and, in doing so, they evolve to derivatives which are the same up to the choice of the parameter K or K'. To express this fact, we have to accurately study how the parameter K is used in a transition $[\![A]\!]K \xrightarrow{\mu} P$. In the proof of Lemma A.5 we use the following fact:

Lemma A.4 *If* $!Q \xrightarrow{\mu} Q'$ *with* $\mu \neq \tau$, *then there exists* Q'' *s.t.* $Q' \equiv Q'' \mid\, !Q$ *and* $Q \xrightarrow{\mu} Q''$. □

Lemma A.5 *Let* A *be a causal term,* $K, K' \subseteq_{\mathrm{fin}} \mathcal{K}$ *and suppose* $[\![A]\!]K \xrightarrow{\mu} P$. *Then*

1. *If* $\mu = a\langle\widetilde{b}h\rangle$ *or* $\mu = (\boldsymbol{\nu}\,\tilde{c}h)\overline{a}\langle\widetilde{b}h\rangle$, *for* $h \notin \mathcal{K}(A) \cup K \cup K'$, *then there exist* $K_0 \subseteq \mathcal{K}(A)$, $\tilde{d}$, A' *and* R *with* $h \notin \mathcal{K}(A')$, *s.t.*

$$P \equiv\ h \rhd (K, K_0)\ \mid (\boldsymbol{\nu}\,\tilde{d})([\![A']\!]K \mid [\![h :: R]\!])$$

and moreover

$$[\![A]\!]K' \xrightarrow{\mu} \equiv\ h \rhd (K', K_0)\ \mid (\boldsymbol{\nu}\,\tilde{d})([\![A']\!]K' \mid [\![h :: R]\!])\,.$$

2. *If* $\mu = \tau$, *then either*
   - *there exists* A' *s.t.* $P \equiv [\![A']\!]K$ *and also* $[\![A]\!]K' \xrightarrow{\tau} Q \equiv [\![A']\!]K'$, *or*
   - *there are* $K_i \subseteq_{\mathrm{fin}} \mathcal{K}(A)$ $i \in \{1, 2\}$,$\tilde{d}$, A', R, *and some* $h \notin (\mathcal{K}(A') \cup K \cup K')$ *s.t.*

$$P \equiv (\boldsymbol{\nu}\,h)\big(\ h \rhd (K_1, K)\ \mid\ h \rhd (K_2, K)\ \mid (\boldsymbol{\nu}\,\tilde{d})([\![A']\!]K \mid [\![h :: R]\!])\big)$$

and also

$$[\![A]\!]K' \xrightarrow{\tau} \equiv (\boldsymbol{\nu}\,h)\big(\ h \rhd (K_1, K')\ \mid\ h \rhd (K_2, K')\ \mid (\boldsymbol{\nu}\,\tilde{d})([\![A']\!]K' \mid [\![h :: R]\!])\big).$$

PROOF: The proof of (1) is by induction on the structure of A. We consider the most significant cases in detail, namely the cases in which the outermost operator of A is a causal prefix or a replication.

Cause prefix $A = H :: B$.

Thus $\llbracket A \rrbracket K = \llbracket B \rrbracket (K, H)$ and $\llbracket A \rrbracket K' = \llbracket B \rrbracket (K', H)$. We have $\llbracket B \rrbracket (K, H) \xrightarrow{\mu} P$. Hence, by the inductive assumption, there are $K'_0 \subseteq \mathcal{K}(B), \widetilde{d}, A''$ and R s.t.

$$\begin{array}{rcl} P & \equiv & h \rhd (K, H, K'_0) \mid (\boldsymbol{\nu}\, \widetilde{d})(\llbracket A'' \rrbracket (K, H) \mid \llbracket h :: R \rrbracket) \\ & = & h \rhd (K, H, K'_0) \mid (\boldsymbol{\nu}\, \widetilde{d})(\llbracket H :: A'' \rrbracket K \mid \llbracket h :: R \rrbracket) \end{array}$$

and

$$\begin{array}{rcl} \llbracket B \rrbracket (K', H) & \xrightarrow{\mu} \equiv & h \rhd (K', H, K'_0) \mid (\boldsymbol{\nu}\, \widetilde{d})(\llbracket A'' \rrbracket (K', H) \mid \llbracket h :: R \rrbracket) \\ & = & h \rhd (K', H, K'_0) \mid (\boldsymbol{\nu}\, \widetilde{d})(\llbracket H :: A'' \rrbracket K' \mid \llbracket h :: R \rrbracket)\,. \end{array}$$

For $K_0 \stackrel{def}{=} H \cup K'_0$ and $A' \stackrel{def}{=} H :: A''$, this concludes the case.

Replication $A = !\, Q$.

Thus $\llbracket A \rrbracket K = !\, \llbracket Q \rrbracket K \xrightarrow{\mu} P$. By Lemma A.4, we have $P \equiv P_1 \mid !\, \llbracket Q \rrbracket K$, for some P_1 s.t. $\llbracket Q \rrbracket K \xrightarrow{\mu} P_1$. Using the inductive assumption on $\llbracket Q \rrbracket K \xrightarrow{\mu} P_1$ we get that for some processes R and S and name tuple $\widetilde{d}$,

$$P_1 \equiv h \rhd K \mid (\boldsymbol{\nu}\, \widetilde{d})(\llbracket S \rrbracket K \mid \llbracket h :: R \rrbracket)$$

and also $\llbracket Q \rrbracket K' \xrightarrow{\mu} P_2$ with

$$P_2 \equiv h \rhd K' \mid (\boldsymbol{\nu}\, \widetilde{d})(\llbracket S \rrbracket K' \mid \llbracket h :: R \rrbracket)\,.$$

Now, from $\llbracket Q \rrbracket K' \xrightarrow{\mu} P_2$ and rule `S-PAR`, we infer

$$\llbracket Q \rrbracket K' \mid !\, \llbracket Q \rrbracket K' \xrightarrow{\mu} P_2 \mid !\, \llbracket Q \rrbracket K' \stackrel{def}{=} P'$$

and hence, applying `S-rep`:

$$!\, \llbracket Q \rrbracket K' = \llbracket !\, Q \rrbracket K' \xrightarrow{\mu} P'\,.$$

To sum up, extruding the scope of $(\boldsymbol{\nu}\, d)$, we have:

$$P \equiv P_1 | !\, \llbracket Q \rrbracket K \equiv h \rhd K \,|(\boldsymbol{\nu}\, \widetilde{d})(\llbracket S \rrbracket K | \llbracket h :: R \rrbracket) | \llbracket !\, Q \rrbracket K \equiv h \rhd K \,|(\boldsymbol{\nu}\, \widetilde{d})(\llbracket S | !\, Q \rrbracket K | \llbracket h :: R \rrbracket)$$

and similarly

$$[\![\,!\,Q]\!]K' \equiv\ h \rhd K'\ \mid (\boldsymbol{\nu}\,\widetilde{d})([\![S \mid\,!\,Q]\!]K' \mid [\![h :: R]\!])\,.$$

For $K_0 \overset{def}{=} \emptyset$ and $A' \overset{def}{=} S \mid\,!\,Q$, this proves our claim.

Now, the proof of assertion (2) of the lemma. The proof is again an induction on the structure of A. The non-trivial cases are when the outermost operator in A is a parallel composition or a replication. We only consider the former case. Thus suppose $A = A_1 \,|\, A_2$; the last rule applied in deriving $[\![A]\!]K \xrightarrow{\tau} P$ must be either a `S-PAR` or a `S-COM`; we analyse the latter case; the former is easier. We have:

$$\texttt{S-COM}:\ \frac{[\![A_1]\!]K \xrightarrow{a\langle\widetilde{b}h\rangle} P_1 \qquad [\![A_2]\!]K \xrightarrow{(\boldsymbol{\nu}\,\widetilde{c}h)\overline{a}\langle\widetilde{b}h\rangle} P_2}{[\![A]\!]K = [\![A_1]\!]K \mid [\![A_2]\!]K \xrightarrow{\tau} (\boldsymbol{\nu}\,\widetilde{c}h)(P_1 \mid P_2) \overset{def}{=} P}\,.$$

For simplicity we suppose that $h \notin K \cup K' \cup \mathcal{K}(A)$; the case $h \in K \cup K' \cup \mathcal{K}(A)$ can be accommodated using Corollary A.3(1) to first rename h to a fresh name.

From part (1) of this lemma, we know that there are $A'_i, K_i \subseteq \mathcal{K}(A_i), \widetilde{d}_i$ and R_i with $h \notin A'_i$, for $i = 1, 2$, such that

$$P_i \equiv\ h \rhd (K_i, K)\ \mid (\boldsymbol{\nu}\,\widetilde{d}_i)([\![A'_i]\!]K \mid [\![h :: R_i]\!])\,.$$

and also

$$[\![A_1]\!]K' \xrightarrow{a\langle\widetilde{b}h\rangle} Q_1 \quad \text{and} \quad [\![A_2]\!]K' \xrightarrow{(\boldsymbol{\nu}\,\widetilde{c}h)\overline{a}\langle\widetilde{b}h\rangle} Q_2 \tag{21}$$

with

$$Q_i \equiv\ h \rhd (K_i, K')\ \mid (\boldsymbol{\nu}\,\widetilde{d}_i)([\![A'_i]\!]K' \mid [\![h :: R_i]\!]),\ \text{for } i \in \{1, 2\}.$$

From the two transitions in (21) and rule `S-COM`, we infer

$$[\![A]\!]K' = [\![A_1]\!]K' \mid [\![A_2]\!]K' \xrightarrow{\tau} (\boldsymbol{\nu}\,\widetilde{c}h)(Q_1 \mid Q_2) \overset{def}{=} Q\,.$$

Finally, defining $A' \overset{def}{=} A'_1 \mid A'_2$, $\widetilde{d} \overset{def}{=} \widetilde{c}\widetilde{d_1}\widetilde{d_2}$ and $R \overset{def}{=} R_1 \mid R_2$, and using the equality $[\![h :: R_1]\!] \mid [\![h :: R_2]\!] = [\![h :: (R_1 \mid R_2)]\!]$, we can write:

$$P \equiv (\boldsymbol{\nu}\,h)\big(\ h \rhd (K_1, K)\ \mid\ h \rhd (K_2, K)\ \mid (\boldsymbol{\nu}\,\widetilde{d})([\![A']\!]K \mid [\![h :: R]\!])\big)$$

and, similarly,

$$Q \equiv (\boldsymbol{\nu}\,h)\big(\ h \rhd (K_1, K')\ \mid\ h \rhd (K_2, K')\ \mid (\boldsymbol{\nu}\,\widetilde{d})([\![A']\!]K' \mid [\![h :: R]\!])\big)\,.$$

This concludes the case. $\square$

We are now ready to prove Lemma 7.1. We first recall its assertion:

$$h \rhd K \mid \llbracket A \rrbracket (h, K_0, K) \lesssim h \rhd K \mid \llbracket A \rrbracket (h, K_0) .$$

Proof of Lemma 7.1. Let

$$H \overset{def}{=} \{h\} \cup K_0 \cup K \qquad \text{and} \qquad H' \overset{def}{=} \{h\} \cup K_0 . \tag{22}$$

We show that the relation

$$\mathcal{R} = \{(h \rhd K \mid \llbracket A \rrbracket H,\ h \rhd K \mid \llbracket A \rrbracket H') \ :\ A \in \mathcal{P}_{c}(\mathcal{N}, \mathcal{K})\}$$

is an expansion up to context and up to $\gtrsim$. Let $(Q_1, Q_2) \in \mathcal{R}$, for $Q_1 \overset{def}{=} h \rhd K \mid \llbracket A \rrbracket H$ and $Q_2 \overset{def}{=} h \rhd K \mid \llbracket A \rrbracket H'$. We show that the moves of Q_1 can be matched by Q_2. The converse, on the actions from Q_2, is proved by reversing the steps. If Q_1 moves, then the move originates either from $h \rhd K$, or from $\llbracket A \rrbracket H$ (an interaction among these two processes is impossible, see Remark 7.12). We only show the details of the second case, since the first case is easier. Thus, we suppose that $Q_1 \xrightarrow{\mu} h \rhd K \mid P_1 \overset{def}{=} Q_1'$, for some P_1 s.t. $\llbracket A \rrbracket H \xrightarrow{\mu} P_1$. We have to find a static context $C[\cdot]$, and processes Q_2', T_1, and T_2 s.t.

$$Q_1' \sim C[T_1], \qquad Q_2 \overset{\mu}{\Longrightarrow} Q_2' \gtrsim C[T_2] \qquad \text{and} \qquad (T_1, T_2) \in \mathcal{R} . \tag{23}$$

We distinguish the cases in which μ is an input, an output, or a silent action.

Case 1: μ is an input, say $\mu = a\langle \widetilde{b} h' \rangle$.

Let h_0 be a fresh name. From $\llbracket A \rrbracket H \xrightarrow{a\langle \widetilde{b} h' \rangle} P_1$ and Corollary A.3(1), we have

$$\llbracket A \rrbracket H \xrightarrow{a\langle \widetilde{b} h_0 \rangle} P_1' \text{ with } P_1'\{h'/h_0\} \equiv P_1 .$$

From Lemma A.5(1), we know that there are $H_0 \subseteq \mathcal{K}(A)$, $\widetilde{d}$ and A' with $h_0 \notin \mathcal{K}(A')$, s.t.

$$P_1' \equiv h_0 \rhd (H_0, H) \mid (\boldsymbol{\nu}\, \widetilde{d})(\llbracket A' \rrbracket H \mid \llbracket h_0 :: R \rrbracket)$$

and moreover

$$\llbracket A \rrbracket H' \xrightarrow{a\langle \widetilde{b} h_0 \rangle} P_2' \equiv h_0 \rhd (H_0, H') \mid (\boldsymbol{\nu}\, \widetilde{d})(\llbracket A' \rrbracket H' \mid \llbracket h_0 :: R \rrbracket) .$$

From the above transition and Corollary A.3(2), we obtain

$$[\![A]\!]H' \xrightarrow{a\langle\widetilde{b}h'\rangle} P_2 \quad \text{with } P_2 \equiv P_2'\{h'/h_0\}.$$

Now, from this, using rule `S-PAR`, we get a transition for Q_2:

$$Q_2 = \ h \rhd K \ \mid [\![A]\!]H' \xrightarrow{a\langle\widetilde{b}h'\rangle} \ h \rhd K \ \mid P_2 \overset{def}{=} Q_2' .$$

Thus, we have found the process Q_2' to use in (23); we have now to find $C[\cdot]$, T_1 and T_2, and prove the requirements in (23). From the definitions of Q_1', P_1 and P_1', we get

$$Q_1' \equiv \ h \rhd K \ \mid P_1'\{h'/h_0\} \equiv \ h \rhd K \ \mid \ h' \rhd (H_0, H) \ \mid (\boldsymbol{\nu}\, \widetilde{d})([\![A']\!]H \mid [\![h' :: R]\!]) \quad (24)$$

and, similarly, from the definitions of Q_2', P_2 and P_2', we get

$$Q_2' \equiv \ h \rhd K \ \mid P_2'\{h'/h_0\} \equiv \ h \rhd K \ \mid \ h' \rhd (H_0, H') \ \mid (\boldsymbol{\nu}\, \widetilde{d})([\![A']\!]H' \mid [\![h' :: R]\!]) . \quad (25)$$

From (24) and (25), extruding the scope of $(\boldsymbol{\nu}\, \widetilde{d})$, and commuting the order of some components, we get

$$\begin{aligned} Q_1' &\equiv (\boldsymbol{\nu}\, \widetilde{d})([\![h' :: R]\!] \mid \ h' \rhd (H_0, H) \ \mid \ h \rhd K \ \mid [\![A']\!]H) && (26)\\ Q_2' &\equiv (\boldsymbol{\nu}\, \widetilde{d})([\![h' :: R]\!] \mid \ h' \rhd (H_0, H') \ \mid \ h \rhd K \ \mid [\![A']\!]H') . && (27)\end{aligned}$$

Recall by (22) that $H = H' \cup K$. Therefore, by Lemma 6.4, we deduce that

$$h' \rhd (H_0, H') \ \mid \ h \rhd K \ \gtrsim \ h' \rhd (H_0, H) \ \mid \ h \rhd K \ .$$

From this and (27) we get

$$Q_2' \gtrsim (\boldsymbol{\nu}\, \widetilde{d})([\![h' :: R]\!] \mid \ h' \rhd (H_0, H) \ \mid \ h \rhd K \ \mid [\![A']\!]H'). \quad (28)$$

Finally, define $C[\cdot] \overset{def}{=} (\boldsymbol{\nu}\, \widetilde{d})([\![h' :: R]\!] \mid \ h' \rhd (H_0, H) \ \mid [\,.\,])$, $T_1 \overset{def}{=} \ h \rhd K \ \mid [\![A']\!]H$ and $T_2 \overset{def}{=} \ h \rhd K \ \mid [\![A']\!]H'$. The results in (26) and (28) show that $Q_1' \equiv C[T_1]$ and that $Q_2' \gtrsim C[T_2]$; this suffices to prove (23), since $\equiv$ implies $\sim$.

Case 2: μ is an output.

This case is similar to the previous one.

Case 3: $\mu = \tau$.

If $[\![A]\!]H \xrightarrow{\tau} P_1$, then, by Lemma A.5(2), one of the following holds:

(a) there is A' s.t. $P_1 \equiv [\![A']\!]H$ and also $[\![A]\!]H' \xrightarrow{\tau} P_2 \equiv [\![A']\!]H'$, or

(b) there are $\tilde{d}$, $A', K_{0i} \subseteq \mathcal{K}(A)$ $(i \in \{1, 2\})$, R and h_0 fresh s.t.

$$P_1 \equiv (\boldsymbol{\nu}\, h_0)\big(\; h_0 \rhd (K_{01}, H) \;\mid\; h_0 \rhd (K_{02}, H) \;\mid (\boldsymbol{\nu}\, \tilde{d})([\![A']\!]H \mid [\![h_0 :: R]\!])\big)$$

and also

$$[\![A]\!]H' \xrightarrow{\tau} P_2 \equiv (\boldsymbol{\nu}\, h_0)\big(\; h_0 \rhd (K_{01}, H') \mid h_0 \rhd (K_{02}, H') \mid (\boldsymbol{\nu}\, \tilde{d})([\![A']\!]H' | [\![h_0 :: R]\!])\big)\,.$$

In either cases, using rule S-PAR, we get

$$Q_2 = \; h \rhd K \;\mid [\![A]\!]H' \xrightarrow{\tau} \; h \rhd K \;\mid P_2 \overset{def}{=} Q_2'\,.$$

If (a) holds, then (23) is immediately validated, since the pair (Q_1', Q_2') belongs to $\mathcal{R}$, up to $\equiv$. Let us consider case (b). By definitions of Q_1', Q_2', P_1 and P_2, we have:

$$\begin{array}{rcl}
Q_1' & = & h \rhd K \;\mid P_1 \\
 & = & h \rhd K \;\mid (\boldsymbol{\nu}\, h_0)\big(\; h_0 \rhd (K_{01}, H) \;\mid\; h_0 \rhd (K_{02}, H) \;\mid (\boldsymbol{\nu}\, \tilde{d})([\![A']\!]H \mid [\![h_0 :: R]\!])\big)
\end{array}$$

and

$$\begin{array}{rcl}
Q_2' & = & h \rhd K \;\mid P_2 \\
 & = & h \rhd K \;\mid (\boldsymbol{\nu}\, h_0)\big(\; h_0 \rhd (K_{01}, H') \;\mid\; h_0 \rhd (K_{02}, H') \;\mid (\boldsymbol{\nu}\, \tilde{d})([\![A']\!]H' \mid [\![h_0 :: R]\!])\big)
\end{array}$$

which, extruding the scope of $\boldsymbol{\nu}\, \tilde{d}$ and $(\boldsymbol{\nu}\, h_0)$, and commuting the order of some components, gives:

$$\begin{array}{l}
Q_1' \equiv \\
\qquad (\boldsymbol{\nu}\, h_0 \tilde{d})\big([\![h_0 :: R]\!] \mid\; h_0 \rhd (K_{01}, H) \;\mid\; h_0 \rhd (K_{02}, H) \;\mid\; h \rhd K \;\mid [\![A']\!]H\big)
\end{array} \tag{29}$$

$$\begin{array}{l}
Q_2' \equiv \\
\qquad (\boldsymbol{\nu}\, h_0 \tilde{d})\big([\![h_0 :: R]\!] \mid\; h_0 \rhd (K_{01}, H') \;\mid\; h_0 \rhd (K_{02}, H') \;\mid\; h \rhd K \;\mid [\![A']\!]H'\big)\,.
\end{array} \tag{30}$$

Since $H = H' \cup K$, by Lemma 6.4 (applied twice), we derive

$$h_0 \rhd (K_{01}, H') \mid h_0 \rhd (K_{02}, H') \mid h \rhd K \;\gtrsim\; h_0 \rhd (K_{01}, H) \mid h_0 \rhd (K_{02}, H) \mid h \rhd K \;\;.$$

From this and (30) we get

$$Q_2' \gtrsim (\boldsymbol{\nu}\, h_0\tilde{d})\big([\![h_0 :: R]\!] \mid\ h_0 \rhd (K_{01}, H)\ \mid\ h_0 \rhd (K_{02}, H)\ \mid\ h \rhd K\ \mid [\![A']\!]H'\big). \quad (31)$$

Finally, define

$$\begin{array}{rcl} C[\cdot] & \overset{def}{=} & (\boldsymbol{\nu}\, h_0\tilde{d})\big([\![h_0 :: R]\!] \mid\ h_0 \rhd (K_{01}, H)\ \mid\ h_0 \rhd (K_{02}, H)\ \mid [\,.\,]\big) \\ T_1 & \overset{def}{=} & h \rhd K\ \mid [\![A']\!]H \\ T_2 & \overset{def}{=} & h \rhd K\ \mid [\![A']\!]H'. \end{array}$$

The results in (29) and (31) show that $Q_1' \equiv C[T_1]$ and that $Q_2' \gtrsim C[T_2]$, thus proving (23).

□

Now Lemma 7.2, whose assertion is:
For any K_0, K_i, $i \in I$, $h \notin K_0 \cup (\cup_{i\in I} K_i)$, A:

$$B \overset{def}{=} (\boldsymbol{\nu}\, h)(\Pi_{i\in I}\ h \rhd K_i\ \mid [\![A]\!](K_0, h))\ \gtrsim [\![A]\!](K_0, \cup_i K_i) \overset{def}{=} A\,.$$

Proof of Lemma 7.2. The proof of this lemma only differ from that of Lemma 7.1 because it uses Lemma 6.6 in place of of Lemma 6.4. □

B Proofs of the Saturation and Cancelation Lemmata

To ease the reading of this appendix, we recall the definitions of Cause-Sets and Saturation.

Definition B.1 (Process cause-sets, Definition 8.1) *If A is a causal process, then $\mathcal{CS}(A)$ is the set of sets of causes inductively defined as follows:*

$$\begin{array}{rcl} \mathcal{CS}(P) & \overset{def}{=} & \{\emptyset\,\} \\ \mathcal{CS}(K :: A) & \overset{def}{=} & \{K \cup H\, | H \in \mathcal{CS}(A)\} \\ \mathcal{CS}(\boldsymbol{\nu}\, a\, A) & \overset{def}{=} & \mathcal{CS}(A) \\ \mathcal{CS}(A_1 \mid A_2) & \overset{def}{=} & \mathcal{CS}(A_1) \cup \mathcal{CS}(A_2)\,. \end{array}$$

Definition B.2 (Saturation, Definition 8.2) *A causal process A is* saturated *for a product of wires $W \overset{def}{=} \Pi_{i \in I}\ k_i \rhd K_i$ if for each $H \in \mathcal{CS}(A)$ and $i \in I$, $k_i \in H$ implies $K_i \subseteq H$.*

To prove the Saturation Lemma, we use Lemma B.3 below on cause-sets of processes. Item (1) says that all sets of causes that appear in causal transitions of a process are also in the cause-sets of that process; item (2) and (3) relate the cause-sets of a process to those of its derivatives; finally, item (4) represents a generalization of item (1) to weak transitions.

Lemma B.3 *Let A be a causal term.*

1. *If for some μ, k, A', $A \xrightarrow[K;k]{\mu} A'$ then $K \in \mathcal{CS}(A)$.*
2. *If $A \xrightarrow[K;k]{\mu} A'$ with $k \notin \mathcal{K}(A)$, and $H \in \mathcal{CS}(A')$, then:*
   - *(a) if $k \notin H$, then $H \in \mathcal{CS}(A)$;*
   - *(b) if $k \in H$, then $H = K \cup \{k\}$.*
3. *Suppose $A \xrightarrow{\tau} A'$ and $H \in \mathcal{CS}(A')$. Then either $H \in \mathcal{CS}(A)$ or $H = H_1 \cup H_2$, with $H_1, H_2 \in \mathcal{CS}(A)$.*
4. *If for some μ, k, A', $A \overset{\mu}{\underset{K;k}{\Longrightarrow}} A'$ then $K = H_1 \cup \cdots \cup H_n$, for some $n > 0$ and $H_1, \ldots, H_n \in \mathcal{CS}(A)$.*

PROOF:

1. A simple transition induction on $A \xrightarrow[K;k]{\mu} A'$.
2. By transition induction on $A \xrightarrow[K;k]{\mu} A'$; we only consider the non-trivial cases, namely PAR and CAU.

   Case a: PAR rule.

   The last rule applied is of the form

$$\text{PAR} : \frac{A_1 \xrightarrow[K;k]{\mu} A_1'}{A = A_1 \mid A_2 \xrightarrow[K;k]{\mu} A_1' \mid A_2 \overset{def}{=} A'} .$$

If $H \in \mathcal{CS}(A') = \mathcal{CS}(A_1') \cup \mathcal{CS}(A_2')$ then, by definition, either $H \in \mathcal{CS}(A_1')$ or $H \in \mathcal{CS}(A_2)$. If $H \in \mathcal{CS}(A_2)$ then $k \notin H$ and, since A_2 is a component of A, also $H \in \mathcal{CS}(A)$. Suppose $H \in \mathcal{CS}(A_1')$. If $k \notin H$ then, by inductive hypothesis, $H \in A_1$. Hence $H \in \mathcal{CS}(A)$ since A_1 is a component of A. Suppose now $k \in H$: By inductive hypothesis, $H = K \cup \{k\}$, that proves the claim.

Case b: `CAU` rule.

The last rule applied is of the form

$$\texttt{CAU}: \frac{A_0 \xrightarrow[K_0;\, k]{\mu} A_0'}{A = K' :: A_0 \xrightarrow[K;\, k]{\mu} K' :: A_0' \overset{def}{=} A'}$$

where $K = K' \cup K_0$. If $H \in \mathcal{CS}(A')$ then, by definition, $H = K' \cup H'$, for some $H' \in \mathcal{CS}(A_0')$. If $k \notin H$, then, by inductive hypothesis, $H' \in \mathcal{CS}(A_0)$ and hence also $H \in \mathcal{CS}(A)$. If $k \in H$, then it must be $k \in H'$. By inductive hypothesis, $H' = K_0 \cup \{k\}$. To sum up, we have: $H = K' \cup H' = K' \cup K_0 \cup \{k\} = K \cup \{k\}$, which is our claim.

3. The proof goes by transition induction. All cases are trivial, except the case when `COM` is the last rule applied, where the last step of the derivation is of the form

$$\texttt{COM}: \frac{A_1 \xrightarrow[K_1;\, k]{\mu_1} A_1',\ A_2 \xrightarrow[K_2;\, k]{\mu_2} A_2'}{A = A_1 \mid A_2 \xrightarrow{\tau} (\boldsymbol{\nu}\, \tilde{c})(A_1'[k \rightsquigarrow K_2] \mid A_2'[k \rightsquigarrow K_1]) \overset{def}{=} A'}$$

with $k \notin \mathcal{K}(A_1, A_2)$. Let $H \in \mathcal{CS}(A_1'[k \rightsquigarrow K_2])$ (the case when $H \in \mathcal{CS}(A_2'[k \rightsquigarrow K_1])$ is perfectly symmetrical). It must be $H = H_0[k \rightsquigarrow K_2]$, for some $H_0 \in \mathcal{CS}(A_1')$. Now, there are two cases, namely $k \notin H_0$ and $k \in H_0$. If $k \notin H_0$, then, from item (2) of this lemma, $H_0 \in \mathcal{CS}(A_1)$ and furthermore $H = H_0$, which proves our claim. If $k \in H_0$, then, from item (2) of this lemma, $H_0 = K_1 \cup \{k\}$; since $k \notin K_1$, this implies $H = K_1 \cup K_2$; but, from item (1) of this lemma, $K_1, K_2 \in \mathcal{CS}(A)$; this proves the claim.

4. Using item (3) of this lemma, one can prove, by induction on m, that

$$\begin{aligned} &\text{if } A \xrightarrow{\tau^m} A',\ m \geq 0, \text{ and } H \in \mathcal{CS}(A'), \text{ then} \\ &H = H_1 \cup \cdots \cup H_n \text{ for some } H_1, \ldots, H_n \in \mathcal{CS}(A). \end{aligned} \tag{32}$$

Now, suppose $A \underset{K;k}{\overset{\mu}{\Longrightarrow}} A'$; this means that $A \xrightarrow{\tau^m} A'' \underset{K;k}{\overset{\mu}{\longrightarrow}} A''' \Longrightarrow A'$, for some m, A'' and A'''. Now, the thesis follows using item (1) of this lemma and (32).

$\square$

We now come to the proof of the Saturation Lemma; we first recall its assertion: Let A be a causal process saturated for the product of wires $\Pi_{i\in I}\, k_i \rhd K_i$. Then:

1. If $A \underset{K;k}{\overset{\mu}{\longrightarrow}} A'$ with k fresh, then A' is saturated for $k \rhd K \mid \Pi_{i\in I}\, k_i \rhd K_i$.
2. If $A \xrightarrow{\tau} A'$, then A' is saturated for $\Pi_{i\in I}\, k_i \rhd K_i$.
3. If $A \underset{K;k}{\overset{\mu}{\Longrightarrow}} A'$ with k fresh, then A' is saturated for $k \rhd K \mid \Pi_{i\in I}\, k_i \rhd K_i$.

Proof of the Saturation Lemma.

1. Let $H \in \mathcal{CS}(A')$ and $h \in H \cap (\{k_i \ : \ i \in I\} \cup \{k\})$. We have to show that $K \subseteq H$ in case $h = k$, and that $K_i \subseteq H$ in case $h = k_i$. This fact follows from Lemma B.3(1) and (2) and from the fact that A is saturated for $\Pi_{i\in I}\, k_i \rhd K_i$.
2. Follows from Lemma B.3(3) and from the definition of saturation.
3. It holds that $A \Longrightarrow A'' \underset{K;k}{\overset{\mu}{\longrightarrow}} A''' \Longrightarrow A'$, for some A'' and A'''; the thesis is obtained by (repeatedly) using item (2) of this lemma on the transition $A \Longrightarrow A''$, then item (1) on the transition $A'' \underset{K;k}{\overset{\mu}{\longrightarrow}} A'''$, then item (2) again (repeatedly) on the transition $A''' \Longrightarrow A'$.

$\square$

We are left with the proof of the Cancelation lemma. For this, we use the following auxiliary technical result:

Lemma B.4 *Let A, B be saturated for $W \overset{def}{=} \Pi_{i\in I}\, k_i \rhd K_i$. Suppose that for some μ*

$$A \underset{H';k_0}{\overset{\mu}{\Longrightarrow}} A' \text{ and } B \underset{H'';k_0}{\overset{\mu}{\Longrightarrow}} B'$$

with k_0 fresh and, furthermore, suppose that

$$W \mid \ k_0 \rhd H' \ \mid [\![A']\!] \approx W \mid \ k_0 \rhd H'' \ \mid [\![B']\!] \,.$$

Then $H' = H''$.

PROOF: We prove that $H' \subseteq H''$ (symmetrically, it will be $H'' \subseteq H'$). Given $k \in H'$, we show that $k \in H''$ as well. Let

$$R \overset{def}{=} W \mid \; k_0 \rhd H' \; \mid [\![A']\!]$$

and

$$S \overset{def}{=} W \mid \; k_0 \rhd H'' \; \mid [\![B']\!].$$

Since $k \in H'$, for all v there is R' s.t.

$$R \xrightarrow{k_0\langle v\rangle} \xrightarrow{\overline{k}\langle v\rangle} R' .$$

Since $R \approx S$, there is S' s.t.

$$S \overset{k_0\langle v\rangle}{\Longrightarrow} \overset{\overline{k}\langle v\rangle}{\Longrightarrow} S' .$$

We now analyse the above sequence of transitions. For some S_1,S_2,S_3,S_4 and S_5 we have

$$S \Longrightarrow S_1 \xrightarrow{k_0\langle v\rangle} S_2 \Longrightarrow S_3 \xrightarrow{\overline{k}\langle v\rangle} S_4 \Longrightarrow S_5 = S' .$$

We now argue on the form of S_1, S_2 and S_3. From Remark 7.12, it follows that process $W \mid \; k_0 \rhd H''$ does not contribute to the weak transition $S \Longrightarrow S_1$; in other words, there exists P s.t. $[\![B']\!] \Longrightarrow P$ and

$$S_1 = W \mid \; k_0 \rhd H'' \; \mid P .$$

From Remark 7.12 and from the fact that $k_0 \notin \{k_i \,:\, i \in I\}$, we get that the transition $S_1 \xrightarrow{k_0\langle v\rangle} S_2$ originates from $k_0 \rhd H''$; thus, letting $Z \overset{def}{=} W \mid \; k_0 \rhd H'' \; \mid \Pi_{k'' \in H''} ! \, \overline{k''}\langle v\rangle$, it holds that

$$S_2 = Z \mid P .$$

Again by Remark 7.12 we know that processes Z and P, as well as any τ-derivative of them, cannot interact with one another. Hence the transition $S_2 \Longrightarrow S_3$ can be decomposed into two independent sequences of transitions from Z and P; formally, there exist V and P' s.t. $Z \Longrightarrow V$, $P \Longrightarrow P'$ and

$$S_3 = V \mid P' .$$

Since $[\![B']\!] \Longrightarrow P'$, from Remark 7.12 it follows that the transition $S_3 \xrightarrow{\overline{k}\langle v\rangle} S_4$ originates from V (i.e. $V \xrightarrow{\overline{k}\langle v\rangle} V'$, for some V'). This and the fact that V is a τ-derivative of Z

imply that $Z \overset{\overline{k}\langle v\rangle}{\Longrightarrow} V'$, and therefore, by definition of Z we get that: Either $k \in H''$, or there is $i \in I$ s.t. $k_i \in H''$ and $k \in K_i$ (so that the transition $\xrightarrow{\overline{k}\langle v\rangle}$ can occur after an interaction of $!\,\overline{k_i}\langle v\rangle$ with $k_i \rhd K_i$). Now, if $k \in H''$ we have finished our proof. Assume therefore that, for some i, $k_i \in H''$ and $k \in K_i$; we show that also $k \in H''$ holds. From Lemma B.3(4) and from $B \overset{\mu}{\underset{H'';k_0}{\Longrightarrow}} B'$, it follows that $H'' = H_1 \cup \cdots \cup H_n$, for some n, with the $H_j \in \mathcal{CS}(B)$, $1 \leq j \leq n$; hence for some $j \in \{1, \ldots, n\}$, we have $k_i \in H_j$; but, since by hypothesis B is saturated for W, it holds that $K_i \subseteq H_j$; since $k \in K_i$ and $H_j \subseteq H''$, we conclude that $k \in H''$, which proves our claim. $\square$

We are now ready to prove the Cancelation Lemma. We first recall its assertion: Let A be saturated for $k \rhd K$ and B be saturated for $k \rhd K'$. Then

$$k \rhd K \mid [\![A]\!] \approx k \rhd K' \mid [\![B]\!] \text{ implies } K = K' \text{ and } [\![A]\!] \approx [\![B]\!]\,.$$

Proof of Cancelation Lemma 8.4. If $k \rhd K \mid [\![A]\!] \approx k \rhd K' \mid [\![B]\!]$, then it must be $K = K'$, otherwise the two processes would be distinguishable through interactions at cause names (by Remark 7.12 processes $[\![A]\!]$, $[\![B]\!]$, and τ-derivatives of them, cannot perform actions at a cause name).

The difficult part is to prove $[\![A]\!] \approx [\![B]\!]$. For this, we show that

$$\begin{array}{rcll} \mathcal{R} & = & \{([\![A\rho]\!], [\![B\rho]\!]) : & \rho \text{ is a finite cause substitution} \\ & & & \text{and for some } I \text{ and } k_i, K_i \quad (i \in I), \\ & & & A \text{ and } B \text{ are saturated for } \Pi_{i\in I}\, k_i \rhd K_i \\ & & & \text{and } \Pi_{i\in I}\, k_i \rhd K_i \mid [\![A]\!] \approx \Pi_{i\in I}\, k_i \rhd K_i \mid [\![B]\!] \quad \} \end{array}$$

is a weak bisimulation up to $\gtrsim$ and up to context. Thus the thesis will follow by letting ρ equal to the empty cause substitution and letting $\Pi_{i\in I}\, k_i \rhd K_i$ equal to $k \rhd K$ in the definition of $\mathcal{R}$. In the rest of the proof, we abbreviate $\Pi_{i\in I}$ as Π_i. We only show how the moves of $[\![A\rho]\!]$ are matched by $[\![B\rho]\!]$, since $\mathcal{R}$ is symmetrical. We consider the case of weak input transitions; the cases of output and invisible transitions are similar or easier. Thus suppose:

$$[\![A\rho]\!] \overset{a(\widetilde{bk})}{\Longrightarrow} P \tag{33}$$

We have to find processes Q, P' and Q' and a static context $C[.]$ s.t.

$$P \gtrsim C[P']\,, \quad [\![B\rho]\!] \overset{a(\widetilde{bk})}{\Longrightarrow} Q \gtrsim C[Q'] \text{ and } (P', Q') \in \mathcal{R}\,. \tag{34}$$

From Proposition 7.11(2), we know that, for some H and A',

$$P \gtrsim \; k \rhd H \; \mid [\![A']\!] \tag{35}$$

and

$$A\rho \overset{a\langle\widetilde{b}\rangle}{\underset{H;k}{\Longrightarrow}} A'. \tag{36}$$

Take a fresh $k_0 \in \mathcal{K}$; since k_0 is fresh, it holds that $k_0 \notin \mathcal{K}(A, B) \cup (\cup_{i \in I} K_i) \cup \{k_i \, : i \in I\}$ and, moreover, $k'\rho = k_0$ iff $k' = k_0$ (note that such a fresh k_0 exists because the nominated sets of causes are finite and ρ is a finite cause substitution).

By applying first Lemma 4.8 and then Lemma 4.7 to (36), we infer that

$$A \overset{a\langle\widetilde{b}\rangle}{\underset{H';k_0}{\Longrightarrow}} A'' \tag{37}$$

with

$$H'\rho = H \text{ and } A''\rho[k_0 \rightsquigarrow k] = A'. \tag{38}$$

Applying again Proposition 7.11 to (37), we get

$$[\![A]\!] \overset{a\langle\widetilde{bk_0}\rangle}{\Longrightarrow} P_1 \gtrsim \; k_0 \rhd H' \; \mid [\![A'']\!] \tag{39}$$

from which, using rule S-PAR, we infer

$$\Pi_i \; k_i \rhd K_i \; \mid [\![A]\!] \overset{a\langle\widetilde{bk_0}\rangle}{\Longrightarrow} \Pi_i \; k_i \rhd K_i \; \mid P_1 \overset{def}{=} R\,. \tag{40}$$

Since, by hypothesis, $\Pi_i \; k_i \rhd K_i \; \mid [\![A]\!] \approx \Pi_i \; k_i \rhd K_i \; \mid [\![B]\!]$, there is S s.t.

$$\Pi_i \; k_i \rhd K_i \; \mid [\![B]\!] \overset{a\langle\widetilde{bk_0}\rangle}{\Longrightarrow} S \approx R\,. \tag{41}$$

Since $\Pi_i \; k_i \rhd K_i$ cannot perform an action at a and, moreover, $\Pi_i \; k_i \rhd K_i$ cannot interact with any T such that $[\![B]\!] \Longrightarrow T$ or $[\![B]\!] \overset{a\langle\widetilde{bk_0}\rangle}{\Longrightarrow} T$ (by Remark 7.12), the term $\Pi_i \; k_i \rhd K_i$ does not contribute to the above weak transition. That is, there exists Q_1 such that

$$[\![B]\!] \overset{a\langle\widetilde{bk_0}\rangle}{\Longrightarrow} Q_1 \tag{42}$$

and

$$S = \Pi_i \; k_i \rhd K_i \; \mid Q_1\,. \tag{43}$$

From (42) and Proposition 7.11(2), we deduce that, for some B'' and H'',

$$B \underset{H'';\,k_0}{\overset{a\langle\widetilde{b}\rangle}{\Longrightarrow}} B'' \tag{44}$$

with

$$Q_1 \gtrsim\ k_0 \rhd H''\ \mid [\![B'']\!]. \tag{45}$$

Now, since $\gtrsim$ implies $\approx$ and parallel composition preserves $\approx$, we have:

$$\begin{array}{rcll}
\Pi_i\ k_i \rhd K_i\ \mid\ k_0 \rhd H'\ \mid [\![A'']\!] & \approx & \Pi_i\ k_i \rhd K_i\ \mid P_1 & \text{(from (39))} \\
& = & R & \text{(from (40))} \\
& \approx & S & \text{(from (41))} \\
& = & \Pi_i\ k_i \rhd K_i\ \mid Q_1 & \text{(from (43))} \\
& \approx & \Pi_i\ k_i \rhd K_i\ \mid\ k_0 \rhd H''\ \mid [\![B'']\!] & \text{(from (45))}\,.
\end{array} \tag{46}$$

Thus, from (37), (44) and (46), we have found that $A \underset{H';\,k_0}{\overset{a\langle\widetilde{b}\rangle}{\Longrightarrow}} A''$, that $B \underset{H'';\,k_0}{\overset{a\langle\widetilde{b}\rangle}{\Longrightarrow}} B''$ and that $\Pi_i\ k_i \rhd K_i\ \mid\ k_0 \rhd H'\ \mid [\![A'']\!] \approx \Pi_i\ k_i \rhd K_i\ \mid\ k_0 \rhd H''\ \mid [\![B'']\!]$; moreover, by definition of $\mathcal{R}$, we know that A and B are saturated for $\Pi_i\ k_i \rhd K_i$. From these facts, using Lemma B.4 we deduce that $H' = H''$.

Having proved that $H' = H''$, we can now conclude the proof of this lemma. Applying Lemma 4.8 and Lemma 4.6 to the transition in (44) we get

$$B\rho \underset{H''\rho;\,k}{\overset{a\langle\widetilde{b}\rangle}{\Longrightarrow}} B'$$

with $B' = B''\rho[k_0 \rightsquigarrow k]$. Since $H' = H''$ and $H'\rho = H$ (by 38), the above transition can be rewritten as

$$B\rho \underset{H;\,k}{\overset{a\langle\widetilde{b}\rangle}{\Longrightarrow}} B'\,.$$

Moreover, by Proposition 7.11(1) we also have

$$[\![B\rho]\!] \overset{a\langle\widetilde{bk}\rangle}{\Longrightarrow} Q \gtrsim\ k \rhd H\ \mid [\![B']\!]\,. \tag{47}$$

We can now show that transition (47) matches the one by $[\![A\rho]\!]$ in (33). For this, we have to exhibit two processes P', Q' and a static context $C[\,.\,]$ s.t. (34) is satisfied. We set $P' \overset{def}{=} [\![A']\!]$, $Q' \overset{def}{=} [\![B']\!]$ and $C[\,.\,] \overset{def}{=}\ k \rhd H\ \mid [\,.\,]$. We first show that $(P', Q') \in \mathcal{R}$. We recall that $A' = A''\rho[k_0 \rightsquigarrow k]$, that $B' = B''\rho[k_0 \rightsquigarrow k]$ and that $\rho[k_0 \rightsquigarrow k]$ is a finite cause substitution. From the Saturation Lemma, applied to the transitions in (37) and (44), and since $H' = H''$, we deduce that A'' and B'' are saturated for $\Pi_i\ k_i \rhd K_i\ \mid\ k_0 \rhd H'$. These facts and (46) demonstrate that $(P', Q') \in \mathcal{R}$. Finally, (35) and (47) demonstrate that $P \gtrsim C[P']$ and $Q \gtrsim C[Q']$; hence (34) holds, which closes up the bisimulation. □