Relating Distributed, Temporal and Causal Observations of Simple Processes*

Luca Aceto
School of Cognitive and Computing Sciences
University of Sussex
Falmer, Brighton BN1 9QH, England
Email: luca@cogs.sussex.ac.uk

Abstract

The noninterleaving equivalences that have been proposed in the literature may be roughly divided into three classes, depending on the kind of observational scenarios underlying them. Some of them distinguish parallelism from nondeterminism by admitting explicit observations of the causal structure of systems; some others are based on the observation of distribution and still others on the assumption that actions have duration.

In general, these three observational scenarios give rise to equivalences of incomparable discriminating power. In this paper, we show that three representative equivalences of the aforementioned classes, namely *timed equivalence* [Hen88], *distributed bisimulation* [CH89] and *causal bisimulation* [DD88], coincide over a language for finite parallel processes without communication and restriction. The proof of this result is algebraic in style and relies on a theorem giving a finite, ω-complete axiomatization of causal bisimulation over the language under consideration. We also give a model for our language based upon causal trees and prove that it is fully abstract with respect to the largest congruence contained in strong bisimulation and preserved by a very simple form of action refinement [AH89].

1 Introduction

In recent years, many notions of equivalence for concurrent systems which distinguish concurrency from nondeterminism in the behaviour of processes have been proposed in the literature, e.g. *timed bisimulation* [Hen88], *NMS bisimulation* [DDNM87], *distributed bisimulation* [CH89], *pomset bisimulation* [BC88], *ST-bisimulation* [GV87], *history preserving bisimulation* [RT88, GG89] and *causal bisimulation* [DD88]. These proposals can be roughly divided into three main classes, depending on the kind of experiments which are used to distinguish parallelism from nondeterminism in the behaviour of processes.

The "causal equivalences", such as history preserving bisimulation and pomset bisimulation, semantically distinguish concurrency from nondeterminism by admitting explicit observations of the *causal structure* of systems, i.e. observations of the actions processes perform during their evolution together with the causal relationships among them. All the equivalences in this class distinguish the CCS processes $p = a|b$ and $q = a.b + b.a$, which are equated by all the interleaving

*The work reported in this paper was mostly carried out while the author was at INRIA-Sophia Antipolis, France, and at the Hewlett-Packard Pisa Science Center, Pisa, Italy.

equivalences proposed in the literature, because q is capable of reacting positively to an experiment of the form "*do* a *causes* b", whilst p can not as in its behaviour a and b are causally independent.

Equivalences like the distributed bisimulations of [CH89] are instead based on the assumption of observers that can detect the distributed nature of the observed systems. In particular, observations of systems will produce *local* and *global* changes in the state of the observed process, which together give the effect of the experimentation. For instance, a possible outcome of an experiment of the form "*do* a" on the process q given above will be to produce the process b as a local residual, i.e. what remains to be executed of the subprocess located at the point of observation. On the other hand, the outcome of the same experiment on the process p will produce the inactive process NIL as local residual. Hence the processes p and q are distinguished by the discipline of observation underlying these distributed equivalences.

Finally, equivalences like timed equivalence and ST-bisimulation equivalence distinguish concurrent from nondeterministic behaviour by taking a more refined view of the level of granularity of actions than the interleaving equivalences. These equivalences are based on the assumption that the actions processes perform during their evolution are not necessarily atomic in time and thus may have duration. In these equivalences, the processes p and q are distinguished because the occurrence of actions a and b may overlap in the behaviour of p, but not in that of q.

Given their intrinsically different nature, it should come as no surprise that these three observational scenarios give rise to semantic equivalences of incomparable discriminating power. (The interested reader is referred to [BCHK91], where several examples supporting this claim are given and explained in detail.) However, in [BCHK91] the authors conjecture that, for a language without communication and restriction, causal and distributed equivalences coincide. In this paper we shall provide a proof of this conjecture. Moreover, we shall also show that, over this restricted subclass of processes, the observational scenario in which actions are assumed to be non-atomic has the same discriminating power of those based upon causality and distribution. Hence, for simple parallel processes, the full power of causal semantics may be obtained by means of the more basic notions of observation based upon duration and distribution.

The proof of the coincidence of causal, distributed and timed equivalence over a language without communication and restriction will be algebraic in style and will rely on showing that these three equivalences have the same axiomatization over the language under consideration. The crux of the proof will be to provide a *finite*, ω-complete axiomatization for causal bisimulation over our simple language, without resorting to the auxiliary prefixing operators used in [DD88]. As argued in [Vaa90], the use of algebraic and axiomatic techniques proves its worth, among other things, in comparing process semantics based on very different notions of observable behaviour of processes.

The paper is organized as follows. Section 2 is devoted to a brief review of background material, to the introduction of the language studied in the paper and of its interleaving semantics. We then present distributed bisimulation equivalence and timed equivalence, together with their algebraic characterizations. Sections 5-6 are devoted to the study of basic properties of causal bisimulation equivalence and to a fully adequate model for it based upon a simple form of Darondeau and Degano's *Causal Trees*. Finally, in § 7 we show that causal bisimulation has the same axiomatization as distributed and timed bisimulation over the language under consideration. As a corollary of this result, we prove that the model given in § 6 is fully abstract with respect to the largest congruence preserved by a very simple form of action refinement [AH89, NEL89] and contained in strong bisimulation equivalence.

2 Preliminaries

In this section, we briefly review some basic notions which will be used throughout the paper and present the language for processes that we will study, together with its interleaving semantics.

Labelled Transition Systems and Bisimulation

In order to make the paper self-contained, we recapitulate some standard material from the literature on process algebra and structural operational semantics [Par81, Mil89, Plo81].

A *labelled transition system* is a structure $\langle \mathsf{St}, \{\stackrel{l}{\longrightarrow} | \ l \in \mathsf{Act}\}\rangle$, where St is a set of states, Act a label set, and, for each $l \in \mathsf{Act}$, $\stackrel{l}{\longrightarrow} \subseteq \mathsf{St} \times \mathsf{St}$ a transition relation.

A *bisimulation* is a symmetric relation $R \subseteq \mathsf{St} \times \mathsf{St}$ such that, for all $(p,q) \in R$ and $l \in \mathsf{Act}$,

$$p \stackrel{l}{\longrightarrow} p' \text{ implies } q \stackrel{l}{\longrightarrow} q' \text{ for some } q' \in \mathsf{St} \text{ such that } (p',q') \in R.$$

The relation of bisimulation equivalence on the set of states St is defined by: $p \sim q$ iff $(p,q) \in R$ for some bisimulation R. The relation $\sim$ is the largest bisimulation over St and it is an equivalence relation.

The Language

The language considered in this paper is essentially a subset of finite CCS, [Mil80, Mil89], enriched with the left-merge operator familiar from the theory of ACP, see e.g. [BK84, BW90]. As pointed out in [Mol89], this operator plays a fundamental role in giving *finite*, complete axiomatizations of bisimulation-like equivalences over process algebras like CCS and ACP. Let A be a countable set of *observable actions* ranged over by a, b, c. The set of processes $\mathbf{P}$ over A is generated by the following grammar:

$$p ::= NIL \mid a.p \mid p + p \mid p \between p \mid p|p$$

where $a \in A$. In what follows, Σ will be used to denote the set of operators occurring in the above BNF-like definition and Σ_2 to denote the set of binary operators in Σ. Following Milner [Mil80, Mil89], NIL is used to denote the inactive process, the process that can not perform any action; $a.p$ denotes a process capable of performing action a and behaving like p thereafter. The nondeterministic composition of processes p and q, $p + q$, is a process that can behave like either one of its components. For processes p and q, $p|q$ is a process that can perform any interleaving of the actions of p and q; on the other hand, $p \between q$ denotes a process which behaves like the parallel composition of p and q, but with the constraint that the first action of $p \between q$ has to be performed by p.

The operational intuition underlying the process combinators described above is formally captured by the axiom and rules in Figure 1. Following Plotkin's SOS [Plo81], these define, for each $a \in A$, a next-state relation $\stackrel{a}{\longrightarrow} \subseteq \mathbf{P} \times \mathbf{P}$ as the least relation which satisfies them. Intuitively, for processes p and q, $p \stackrel{a}{\longrightarrow} q$ is interpreted as "process p can perform action a and behave like q thereafter". The operational semantics for the language $\mathbf{P}$ is then given by the labelled transition system $\langle \mathbf{P}, \{\stackrel{a}{\longrightarrow} | \ a \in A\}\rangle$. The largest bisimulation over this LTS, denoted by $\sim$, will be referred to as *bisimulation equivalence*.

Bisimulation equivalence is an attempt at formalizing the idea that two processes are indistinguishable when they show essentially the same interactive behaviour. As it is well-known, see e.g. [BK84, Mil89], it is an equivalence relation and a Σ-congruence.

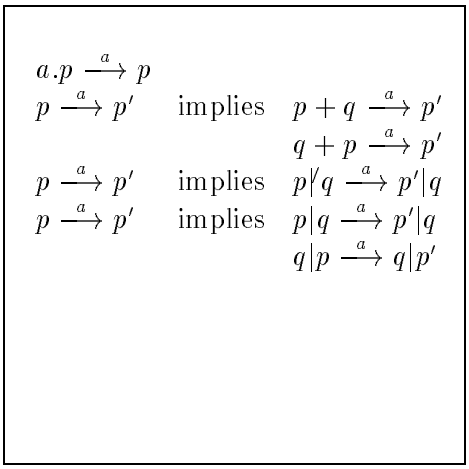

Figure 1: Axiom and rules for $\xrightarrow{a}$

One of the assumptions upon which the theory of standard bisimulation equivalence is based is that processes evolve by performing actions which are atomic both in space and time. (The interested reader is referred to the introduction to chapter 6 of [Cas88] for a clear discussion of this issue.) As a consequence, parallelism is not a primitive notion in its semantic theory, in the sense that, at least for finite processes, one can always find a purely sequential process which is equivalent to a parallel one. A standard example of the semantic reduction of parallelism to sequential nondetermism performed by $\sim$ is the equivalence

$$a.b.NIL + b.a.NIL \quad \sim \quad a.NIL|b.NIL \tag{1}$$

which shows that a process that can perform actions a and b in parallel is equated to one that can perform them in any temporal order. This observation has led several researchers to investigate semantic theories for process description languages in which actions are no longer atomic in space or time, see e.g. [CH89, Hen88, GV87]. In the next two sections, we shall briefly review the basic ideas underlying two variants of bisimulation equivalence: the *distributed bisimulation equivalence* proposed in [CH89] and the *timed equivalence* of [Hen88] (also known as *split-2 equivalence* [GV87]). We shall not go into the details of the theories of these two equivalences. In fact, all we shall need to know about them is that they have been equationally characterized over the language **P**. We refer the interested reader to the afore-mentioned references for more details.

3 A Semantics Based on the Observation of Distribution

Distributed bisimulation equivalence, as proposed in [CH89], is a variation on bisimulation equivalence based upon the idea that distribution is an observable property of systems. In this view, a term of the form $p|q$ describes a process consisting of two physically separated subprocesses p and q. The observational framework underlying it is so described in [CH89]:

> ...each observer is placed at some definite – but unspecified – locality; at each step he can ask for an action from the component at his locality, and observe the *local* result of this action. At the same time, he is informed of the *global* result of his observation.

This intuition is formally described by transitions of the form

$$p \xrightarrow{a}_d \langle q, r \rangle$$

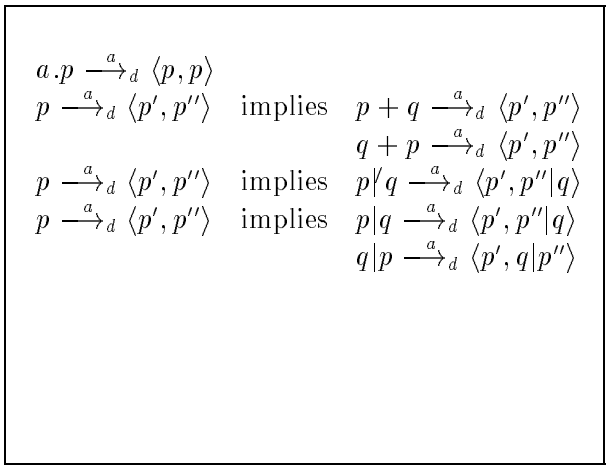

Figure 2: Axiom and rules for $\xrightarrow{a}_d$

which are interpreted as saying that "process p can react to an a-experiment, and that this experiment changes the local component at the point of observation into q and the whole process into r" [CH89]. These distributed transitions are defined for the language $\mathbf{P}$ by the SOS rules given in Figure 2. Comments on them may be found in the above-given reference.

A *distributed bisimulation* is a symmetric relation $R \subseteq \mathbf{P} \times \mathbf{P}$ which satisfies, for every $(p, q) \in R$ and $a \in A$, the following property:

$$p \xrightarrow{a}_d \langle p', p''\rangle \text{ implies } q \xrightarrow{a}_d \langle q', q''\rangle \text{ for some } q', q'' \text{ such that } (p', q') \in R \text{ and } (p'', q'') \in R.$$

Then processes p and q are distributed bisimulation equivalent, written $p \sim_d q$, iff $(p, q) \in R$ for some distributed bisimulation R. As previously mentioned, in the remainder of this paper we shall not need to know much of the rich theory of distributed bisimulation equivalence developed in [CH89]. In fact, our comparison of this equivalence with the timed equivalence proposed in [Hen88] and the causal equivalence of [DD88] will only rely on the equational characterization of $\sim_d$ which is given in the following theorem.

Theorem 3.1 (Castellani and Hennessy [CH89]) *For all $p, q \in \mathbf{P}$, $p \sim_d q$ iff $p =_E q$, where $=_E$ denotes the least Σ-congruence which satisfies the equations in Figure 3.*

4 A Semantics Based on the Observation of Duration

In this section we shall briefly review the theory of timed bisimulation equivalence proposed in [Hen88] and give a sketch of the proof of an equational characterization of this equivalence over the language $\mathbf{P}$.

Timed bisimulation equivalence is just a standard bisimulation equivalence, but based upon the idea that processes evolve by performing beginnings and endings of actions and that these subactions may be observed. Formally, following [Hen88], we shall use $S(a)$ and $F(a)$ to denote the start and the end of action $a \in A$, respectively, and $A_s = \{S(a), F(a) \mid a \in A\}$ to denote the set of these subactions. The set A_s will be ranged over by α.

In order to give a timed operational semantics for the language $\mathbf{P}$, we enlarge the language by considering prefixing operators of the form $F(a)$._, one for each $a \in A$. Intuitively, in state $F(a).p$

$$
\begin{array}{lrcl}
(A1) & x+(y+z) & = & (x+y)+z \\
(A2) & x+y & = & y+x \\
(A3) & x+x & = & x \\
(A4) & x+NIL & = & x \\
 & & & \\
(LM1) & (x+y)\between z & = & x\between z+y\between z \\
(LM2) & (x\between y)\between z & = & x\between(y|z) \\
(LM3) & x\between NIL & = & x \\
(LM4) & NIL\between x & = & NIL \\
 & & & \\
(PAR) & x|y & = & x\between y+y\between x
\end{array}
$$

Figure 3: The set of equations E

the execution of action a has started, but not yet terminated. Formally, the set $\mathcal{S}$ ($s, s' \in \mathcal{S}$) of *process states* is given by the grammar

$$s ::= NIL \mid a.s \mid F(a).s \mid s+s \mid s|s \mid s\between s.$$

Note that $\mathbf{P}$ is a sub-language of $\mathcal{S}$. For each subaction $\alpha \in A_s$, $\stackrel{\alpha}{\longrightarrow}_t$ will be used to denote the least binary relation over $\mathcal{S}$ which satisfies the axioms and rules in Figure 4. The interested reader is referred to [Hen88, AH89] for comments on these rules. The largest bisimulation over the LTS $\langle \mathcal{S}, \{\stackrel{\alpha}{\longrightarrow}_t \mid \alpha \in A_s\}\rangle$ will be denoted by $\sim_t$ and will be called *timed bisimulation equivalence*. In what follows, we shall only be interested in the identifications induced by this equivalence over the language $\mathbf{P}$.

The following theorem, which gives an equational characterization of the equivalence $\sim_t$ over the language $\mathbf{P}$, will be all we shall need to know about $\sim_t$ in the remainder of the paper. The proof of the "completeness" implication of this theorem relies, as usual, on the isolation of suitable normal forms for processes. These take the form

$$\sum_{i\in I} a_i.p_i\between p'_i$$

where I is a finite index set and p_i, p'_i are themselves normal forms. The proof of the following normalization result is standard; see e.g. Lemma 1.3.5 of [CH89].

Lemma 4.1 *For every $p \in \mathbf{P}$ there exists a normal form q such that $p =_E q$.*

Theorem 4.1 (Completeness for $\sim_t$) *For all $p, q \in \mathbf{P}$, $p \sim_t q$ iff $p =_E q$.*

Proof: (Sketch) The soundness of the axioms in Figure 3 with respect to $\sim_t$ is easily checked. In view of Lemma 4.1 and of the soundness of the equations in E with respect to $\sim_t$, it is sufficient to prove completeness for normal forms. Following the proof of Theorem 2.2.9 in [Hen88], we can prove that the equations in E suffice to prove any two equivalent normal forms equal. The proof of this result makes a fundamental use of a simplified version of Proposition 2.2.8 of [Hen88]. □

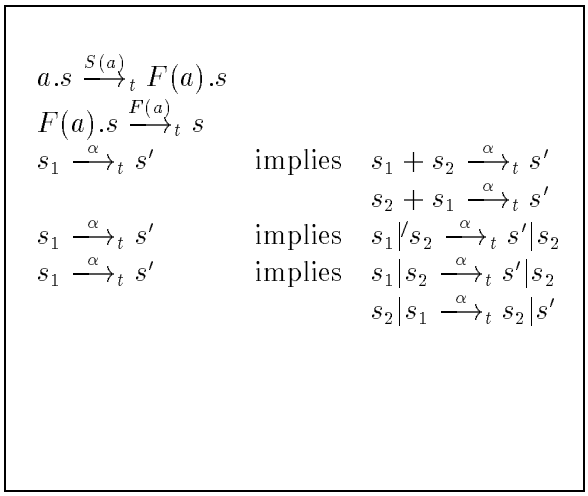

Figure 4: Axioms and rules for $\xrightarrow{\alpha}_t$

5 A Semantics Based on the Observation of Causality

As mentioned in the introduction, notions of semantic equivalence which distinguish concurrency from nondeterminism in the behaviour of processes may be obtained by means of observational scenarios based on experimentations on the causal structure of systems. In this section, we shall present a semantics based upon this idea for the language **P** following Darondeau and Degano's work on *Causal Trees*, see e.g. [DD88, DD90]. In particular, we shall endow the language **P** with a causal operational semantics similar to the one presented in [DD88] for full CCS and study the resulting notion of bisimulation. The causal equivalence obtained in this way will be shown to coincide with $\sim_d$ and $\sim_t$ over the language **P**; this shows that the full power of causal semantics over the language **P** can be obtained by means of the more basic notions of observation underlying $\sim_d$ and $\sim_t$. Our results will also enable us to relate the semantic model of causal trees to the testing scenario underlying standard bisimulation equivalence. In fact, we shall show that a denotational semantics for the language **P** in terms of a simple form of causal trees is the coarsest one compatible with strong bisimulation equivalence in all refinement contexts. (See [Ace91], where a similar result was established for series-parallel pomsets.)

Due to the simplicity of the causal relationships among actions which may be exhibited by the processes in **P**, it will be possible to give a causal semantics for our language in terms of a rather simple notion of causal tree.

Definition 5.1 (Simple Causal Trees) *A* simple causal tree *is a finite synchronization tree T* [Mil80] *labelled on L ($l, l' \in L$), where*

$$L = \{(a, K) \mid a \in A \text{ and } (K = \varnothing \text{ or } K = \{n\}, \text{ for some } n \in I\!N_+)\}.$$

We use $I\!N_+$ to denote the set of positive integers.

Following [DD88], simple causal trees will be considered modulo bisimulation equivalence and will be denoted by terms over the obvious syntax for L-labelled trees. Formally, the set SCT of simple causal trees is generated by the inductive definition

$$\frac{\{l_i \in L, T_i \in \mathsf{SCT} \mid i \in I\} \ (I \text{ a finite index set})}{\sum_{i \in I} l_i.T_i \in \mathsf{SCT}}$$

By convention, $\sum_{i \in \varnothing} l_i.T_i =_{def} 0$, where 0 denotes the empty tree. The sum operation over SCT is assumed to be associative, commutative and idempotent; moreover, it has 0 as neutral element.

Intuitively, causal trees are just synchronization trees over an enriched set of labels, which carry not only information about what actions have occurred, but also about the causal dependencies amongst them. Each label (a, K) will be used to denote the occurrence of action a with K being a set of backward pointers to the arcs in the tree which stand for the *direct* causes of a. Due to the simplicity of the language under consideration, the set of direct causes of an action a will always turn out to be empty or a singleton. For this reason, the class of simple causal trees introduced in the previous definition will be sufficiently rich to provide a causal semantics for $\mathbf{P}$.

In order to provide a causal operational semantics for $\mathbf{P}$ à la Darondeau and Degano (and following their example), we shall embed the language for processes into the language $\mathbf{P}_c$ generated by the following grammar:

$$e ::= (K \Rightarrow p) \mid e + e \mid e \not| e \mid e|e,$$

where $K = \varnothing$ or $K = \{n\}$ for some $n \in I\!N_+$. The operators $(K \Rightarrow _)$ will be used to indicate, at each step, "the activating causes of all the active terms and subterms, given by backward references to the past of the derivation" [DD88]. A process $p \in \mathbf{P}$ will be identified with the term $(\varnothing \Rightarrow p) \in \mathbf{P}_c$. Intuitively, the causal semantics of process p will be obtained by letting p evolve by giving it an empty set of direct causes.

Following [DD88], each term $e \in \mathbf{P}_c$ will be considered in canonical form with respect to the operators $(K \Rightarrow _)$. In particular, it will be assumed that the operators $(K \Rightarrow _)$ distribute over all the binary operators in Σ_2. Formally, let $\equiv$ denote the least Σ_2-congruence over $\mathbf{P}_c$ which satisfies, for each $\odot \in \Sigma_2$, the following axiom:

$$(D) \quad (K \Rightarrow (p \odot q)) \quad = \quad (K \Rightarrow p) \odot (K \Rightarrow q).$$

Applying axiom (D) as a rewrite rule from left to right, it is easy to see that for each $e \in \mathbf{P}_c$ there exists a *canonical term* $\gamma(e)$ generated by the grammar

$$\gamma ::= (K \Rightarrow NIL) \mid (K \Rightarrow a.p) \mid \gamma + \gamma \mid \gamma \not| \gamma \mid \gamma | \gamma$$

such that $e \equiv \gamma(e)$. In what follows, $\mathbf{P}_c$ will always be considered modulo $\equiv$. We shall use Σ' to denote the extension of the signature Σ with the unary operators $(K \Rightarrow _)$.

Notation 5.1 *Following Darondeau and Degano, the notation $\delta(K \Rightarrow p)$ is used as a short-hand for $(\{n+1 \mid n \in K\} \Rightarrow p)$. This notation is extended to the whole of $\mathbf{P}_c$ by assuming that $\delta(\cdot)$ distributes over all the operators in Σ_2. See* [DD88] *for a discussion of the intuition underlying $\delta(\cdot)$.*

We shall now define a Plotkin-style, [Plo81], causal operational semantics for the language $\mathbf{P}_c$, and hence for $\mathbf{P}$ via the identification of $p \in \mathbf{P}$ with $(\varnothing \Rightarrow p) \in \mathbf{P}_c$. For each $l \in L$, let $\stackrel{l}{\longrightarrow}_c$ denote the least binary relation over $\mathbf{P}_c$ which satisfies the axiom and rules in Figure 5. The interested reader is referred to [DD88] for detailed motivations and comments on these rules.

As an example of the application of the operational rules given in Figure 5, let us consider two computations associated with the terms $a.b.NIL + b.a.NIL$ and $a.NIL|b.NIL$.

$$(\varnothing \Rightarrow (a.b.NIL + b.a.NIL)) \equiv (\varnothing \Rightarrow a.b.NIL) + (\varnothing \Rightarrow b.a.NIL) \stackrel{(a,\varnothing)}{\longrightarrow}_c (\{1\} \Rightarrow b.NIL)$$
$$\stackrel{(b,\{1\})}{\longrightarrow}_c (\{1\} \Rightarrow NIL).$$

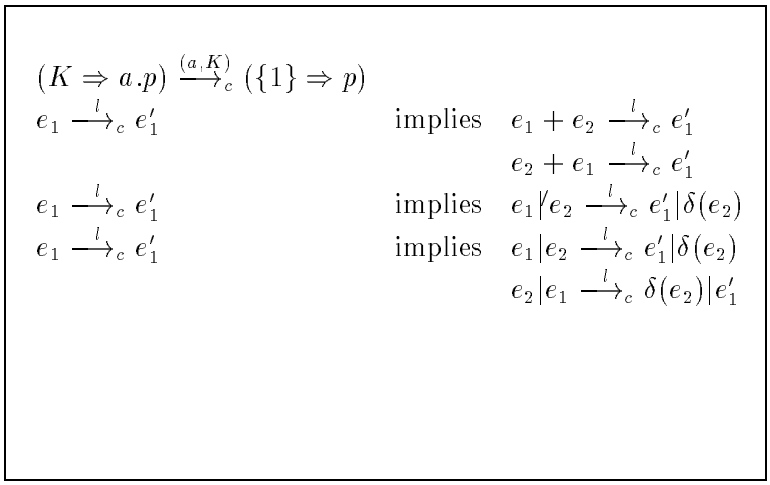

Figure 5: Axiom and rules for $\xrightarrow{l}_c$

In the sequence of transitions given above, action b occurs with $\{1\}$ as set of backward pointers to its set of direct causes. In this computation, the pointer 1 refers to the previous occurrence of action a. Thus, *a causes b* in this computation of $(\varnothing \Rightarrow (a.b.NIL + b.a.NIL))$. On the other hand, actions a and b are independent of each other in any computation of $a.NIL|b.NIL$. This is reflected in the causal operational semantics by the fact that both actions will have an empty set of pointers to their direct causes. For example,

$$(\varnothing \Rightarrow (a.NIL|b.NIL)) \equiv (\varnothing \Rightarrow a.NIL)|(\varnothing \Rightarrow b.NIL) \quad \begin{array}{ll} \xrightarrow{(a,\varnothing)}_c & (\{1\} \Rightarrow NIL)|(\varnothing \Rightarrow b.NIL) \\ \xrightarrow{(b,\varnothing)}_c & (\{1\} \Rightarrow NIL)|(\{1\} \Rightarrow NIL). \end{array}$$

The above example shows the "expressiveness" of the causal operational semantics. In fact, in this semantics there is a whole set of transitions which allows us to differentiate the processes $a.b.NIL + b.a.NIL$ and $a.NIL|b.NIL$, which are identified by all the interleaving equivalences proposed in the literature.

Following Park and Milner [Par81, Mil89], we shall now define a notion of semantic equivalence over the language $\mathbf{P}_c$, and consequently on the process language $\mathbf{P}$, by means of the bisimulation technique.

Definition 5.2 (Causal Equivalence) *$\sim_c$ will denote the largest bisimulation over the labelled transition system $\langle \mathbf{P}_c, \{\xrightarrow{l}_c | \ l \in L\}\rangle$. The equivalence $\sim_c$ will be called* causal equivalence. *For all $p, q \in \mathbf{P}$,*

$$p \sim_c q \Leftrightarrow_{def} (\varnothing \Rightarrow p) \sim_c (\varnothing \Rightarrow q).$$

The following are some basic properties of $\sim_c$ which will be used throughout the paper.

Definition 5.3 *For each $e \in \mathbf{P}_c$, $\| e \|$ will denote the length of the longest sequence of actions in L that e may perform. Formally,*

$$\| e \| = max\{n \mid \text{ there exist } l_1, \ldots, l_n \in L, e_1, \ldots, e_n \in \mathbf{P}_c \text{ such that } e \xrightarrow{l_1}_c e_1 \xrightarrow{l_2}_c \cdots \xrightarrow{l_n}_c e_n\}.$$

For all $p \in \mathbf{P}$, $\| p \| =_{def} \| (\varnothing \Rightarrow p) \|$.

The following properties of $\sim_c$ and $\| \cdot \|$ are easily seen to hold:

Fact 5.1

(1) $|$ *is commutative and associative modulo* $\sim_c$.

(2) *For all* $e_1, e_2 \in \mathbf{P}_c$, $e_1 \sim_c e_2$ *implies* $\| e_1 \| = \| e_2 \|$.

(3) *For all* $e_1, e_2 \in \mathbf{P}_c$, $\| e_1 \| = 0$ *implies* $e_1 | e_2 \sim_c e_2$.

The remainder of this paper will be mainly devoted to showing that, over the language $\mathbf{P}$, causal equivalence $\sim_c$ coincides with the distributed bisimulation equivalence $\sim_d$ introduced in [CH89], and briefly discussed in § 3, and with the timed-equivalence proposed in [Hen88]. (See also § 4.) The proof of the announced result will be algebraic in style and exploits the equational characterizations for $\sim_d$ and $\sim_t$ presented in § 3 and § 4, respectively. In fact, Theorems 3.1 and 4.1 give us a handle for proving the equivalence of causal, distributed and timed equivalence over $\mathbf{P}$. It is sufficient to show that $\sim_c$ over $\mathbf{P}$ is completely axiomatized by the set of equations E in Figure 3. This will be the subject of the following sections.

6 A Model for $\sim_c$ in Terms of Simple Causal Trees

In order to study some of the properties of $\sim_c$ over the language $\mathbf{P}$ it will be convenient to have a tree-model for this language which is in complete agreement with the relation of causal equivalence. This will allow us to prove properties of $\sim_c$ over the denotational model rather than in terms of the causal operational semantics. Following [DD88], a key application of this approach will be in proving that $\sim_c$ is a congruence over $\mathbf{P}$.

The model for the language $\mathbf{P}$ studied in this section is a variation on the causal trees ones studied in [DD88] and we shall closely follow this reference in the investigation of some of its properties. In order to establish a link with the behavioural semantics, we shall interpret the language $\mathbf{P}$ over a suitable algebra of simple causal trees. (See Definition 5.1.)
In order to give an interpretation of the language $\mathbf{P}$ over the set of simple causal trees SCT it will be necessary to give SCT the algebraic structure underlying $\mathbf{P}$ and, more generally, $\mathbf{P}_c$. In particular, we shall have to define over SCT the combinators used in the formation rules for these languages. The definition of these operators over SCT will make use of some auxiliary operators.

The following definitions are essentially borrowed from [DD88], but are considerably simpler than those given in that reference because of the simplicity of the causal dependencies expressible in SCT. We refer the interested reader to [DD88] for the intuition underlying the definitions to follow and more information on the nature of models for CCS-like languages given in terms of causal trees.

Definition 6.1 (The Auxiliary Operators $\langle n \rangle$ and $[n]$) *For each $n \in I\!N_+$, the auxiliary operators $\langle n \rangle$ and $[n]$ are defined by induction on the depth of the simple causal tree $T = \sum_{i \in I}(a_i, K_i).T_i$ as follows:*

$$\langle n \rangle T = \sum_{i \in I}(a_i, \langle n \rangle K_i).\langle n+1 \rangle T_i \quad \textit{where } \langle n \rangle K_i =_{def} \begin{cases} \{n\} & \textit{if } K_i = \varnothing \\ K_i & \textit{otherwise} \end{cases}$$

$$[n] T = \sum_{i \in I}(a_i, [n] K_i).[n+1] T_i \quad \textit{where } [n] K_i =_{def} \begin{cases} \{m+1\} & \textit{if } K_i = \{m\} \textit{ and } m \geq n \\ K_i & \textit{otherwise.} \end{cases}$$

Operators $[n]$ will be used to increment all the backward pointers in the tree T greater than or equal to n. In particular, the cause incrementing operation [1] will be used as a semantic counterpart of

the δ operator used in the definition of the operational semantics for $\mathbf{P}_c$. Operators $\langle n \rangle$ will be the semantic counterparts of the $(\{n\} \Rightarrow _)$ operations and will also find application in the definition of the prefixing operation over SCT. (See Definition 6.3.)

In the technical results which will be established in the remainder of this section, we shall make use of some combinatorial properties of the operators $\langle n \rangle$ and $[n]$. The following fact can be easily shown by induction on the depth of the tree T:

Fact 6.1 *For all $T \in \mathsf{SCT}$, the following properties hold:*

(1) *for each $n, m \in \mathbb{N}_+$, $\langle n \rangle(\langle m \rangle T) = \langle m \rangle T$,*

(2) *for each $n \in \mathbb{N}_+$, $[n+1]([n]T) = [n]([n]T)$.*

We shall now establish a proposition which will play a fundamental rôle in relating the syntactic operator δ with the semantic one [1] defined above. For the sake of completeness, we recall here the following key definition from [DD88].

Definition 6.2 (Tree-Transitions & Bounded Causal Trees)

(1) *Let $T = \sum_{i \in I} l_i.T_i$ and U be simple causal trees. We write $T \stackrel{l}{\twoheadrightarrow} U$ iff there exists $i \in I$ such that $l_i = l$ and $T_i = U$. For each $\omega \in L^\star$, $\stackrel{\omega}{\twoheadrightarrow}$ is defined in the standard way.*

(2) *A causal tree T is n-*bounded*, $n \in \mathbb{N}_+$, iff for all $\omega \in L^\star$, $(a, K) \in L$, $T \stackrel{\omega.(a,K)}{\twoheadrightarrow}$ implies $m \leq n + |\omega|$, for all $m \in K$. Here $|\omega|$ denotes the length of the sequence ω.*

Note that if $T = \sum_{i \in I} l_i.T_i$ is n-bounded then each subtree T_i is $(n+1)$-bounded. This observation will enable us to give an inductive proof of the following useful property of trees which are n-bounded, for some $n \in \mathbb{N}_+$.

Proposition 6.1 *For each $T \in \mathsf{SCT}$ and $n \in \mathbb{N}_+$, if T is n-bounded then $[n+1](\langle n \rangle T) = \langle n \rangle T$.*

Proof: See Appendix A. □

The following fact collects two more combinatorial properties of the operators $\langle n \rangle$ and $[n]$ which will find application in what follows.

Fact 6.2 *For all $T \in \mathsf{SCT}$, $n, m \in \mathbb{N}_+$, the following properties hold:*

(1) $\langle n+m \rangle [m]T = [m]\langle (n+m)-1 \rangle T$ *and*

(2) $[n+m][m]T = [m][(n+m)-1]T$.

Proof: See Appendix A. □

We can now introduce the Σ'-algebra of simple causal trees we shall be working with in the remainder of this section.

Definition 6.3 *The set of simple causal trees SCT is turned into a Σ'-algebra by interpreting NIL as the empty tree 0, $+$ as the sum of trees and defining the other operators as follows:*

$$\begin{aligned}
(\varnothing \Rightarrow T) &= T \\
(\{n\} \Rightarrow T) &= \langle n \rangle T \\
a.T &= (a, \varnothing).\langle 1 \rangle T \\
T \between U &= \textstyle\sum_{i \in I} l_i.(T_i | [1]U) \\
T | U &= T \between U + U \between T,
\end{aligned}$$

where $T, U \in \mathsf{SCT}$ and $T = \sum_{i \in I} l_i.T_i$.

With abuse of notation, $[\![\cdot]\!]$ will be used to denote both the unique Σ-homomorphism from $\mathbf{P}$ onto SCT and the unique Σ'- homomorphism from $\mathbf{P}_c$ onto SCT. Our aim is to prove that, for all $e_1, e_2 \in \mathbf{P}_c$,

$$e_1 \sim_c e_2 \quad \text{iff} \quad [\![e_1]\!] = [\![e_2]\!]. \tag{2}$$

Among other things, this will entail that, for processes $p, q \in \mathbf{P}$,

$$p \sim_c q \quad \text{iff} \quad [\![(\varnothing \Rightarrow p)]\!] = [\![(\varnothing \Rightarrow q)]\!] \quad \text{iff} \quad [\![p]\!] = [\![q]\!].$$

Following [DD88], in order to prove that the operational and denotational models for the language $\mathbf{P}_c$ are in complete agreement, we shall establish two results showing that the transition system semantics for $e \in \mathbf{P}_c$ defined in § 5 coincides with the set of tree-transitions for $[\![e]\!]$. This amounts to proving that, for each $e \in \mathbf{P}_c$,

- $e \stackrel{l}{\longrightarrow}_c e'$ implies $[\![e]\!] \stackrel{l}{\twoheadrightarrow} [\![e']\!]$ and
- $[\![e]\!] \stackrel{l}{\twoheadrightarrow} x$ implies $e \stackrel{l}{\longrightarrow}_c e'$ for some e' such that $[\![e']\!] = x$.

Having established these two results, it will then be possible to prove (2). In relating operational and tree-transitions for $e \in \mathbf{P}_c$, it will be necessary to establish a connection between the syntactic δ operator and the operator $[1]$ over SCT. More precisely, we shall aim at proving that, for each $e \in \mathbf{P}_c$,

$$[\![\delta(e)]\!] \;=\; [1][\![e]\!]. \tag{3}$$

In the proof of (3), we shall make use of several technical lemmas. First of all, we shall prove a result expressing the interplay between $[1]$ and the binary operators in the signature for processes Σ.

Lemma 6.1 *For all $T, U \in \mathsf{SCT}$ and $n \in \mathbb{N}_+$, the following equalities hold:*

(1) $[n](T + U) = [n]T + [n]U$,

(2) $[n](T \between U) = [n]T \between [n]U$ *and*

(3) $[n](T|U) = [n]T|[n]U$.

Proof: See Appendix A. □

We shall now show that, for each $p \in \mathbf{P}$, $[\![p]\!]$ is a 1-bounded, simple causal tree. This result will allow us to use Fact 6.1 in the proof of property (3).

Lemma 6.2 *For each $p \in \mathbf{P}$, $[\![p]\!]$ is a 1-bounded simple causal tree.*

Proof: See Appendix A. □

The next step towards a proof of property (3) is to show that the interpretation of $\mathbf{P}_c$ in terms of SCT given in Definition 6.3 satisfies the relation $\equiv$, i.e. that for all $e_1, e_2 \in \mathbf{P}_c$, $e_1 \equiv e_2$ implies $[\![e_1]\!] = [\![e_2]\!]$. The main import of this result is that in proving statements about trees of the form $[\![e]\!]$, we may safely assume that e is in canonical form. (See page 8 for a definition of this notion.)

In order to prove that the Σ'-algebra of SCT trees satisfies $\equiv$, it is sufficient to check that, for all $e_1, e_2 \in \mathbf{P}_c$, the following equalities hold:

$$\begin{array}{lrcl}
(D1) & [\![(K \Rightarrow (e_1 + e_2))]\!] & = & [\![(K \Rightarrow e_1)]\!] + [\![(K \Rightarrow e_2)]\!] \\
(D2) & [\![(K \Rightarrow (e_1 \between e_2))]\!] & = & [\![(K \Rightarrow e_1)]\!] \between [\![(K \Rightarrow e_2)]\!] \\
(D3) & [\![(K \Rightarrow (e_1 | e_2))]\!] & = & [\![(K \Rightarrow e_1)]\!] | [\![(K \Rightarrow e_2)]\!].
\end{array}$$

The verification of equalities (D1)-(D3) is immediate if $K = \emptyset$. In what follows we shall prove that they also hold when $K = \{n\}$. The following lemma will be useful in the proofs of equalities (D2) and (D3).

Lemma 6.3 *For all $T, U \in \mathsf{SCT}$, $n \in I\!N_+$, $\langle n \rangle (T|U) = \langle n \rangle T | \langle n \rangle U$.*

Proof: See Appendix A. □

Fact 6.3 *For all $e_1, e_2 \in \mathbf{P}_c$, $n \in I\!N_+$, the following equalities hold:*

(1) $[\![(\{n\} \Rightarrow (e_1 + e_2))]\!] = [\![(\{n\} \Rightarrow e_1)]\!] + [\![(\{n\} \Rightarrow e_2)]\!]$;

(2) $[\![(\{n\} \Rightarrow (e_1 \between e_2))]\!] = [\![(\{n\} \Rightarrow e_1)]\!] \between [\![(\{n\} \Rightarrow e_2)]\!]$;

(3) $[\![(\{n\} \Rightarrow (e_1 | e_2))]\!] = [\![(\{n\} \Rightarrow e_1)]\!] \mid [\![(\{n\} \Rightarrow e_2)]\!]$.

Proof: Equality (1) is immediate and (3) follows from Lemma 6.3. Equality (2) can be checked by calculation using Fact 6.2(1) and Lemma 6.3. The details are omitted. □

Corollary 6.1 *For all $e_1, e_2 \in \mathbf{P}_c$, $e_1 \equiv e_2$ implies $[\![e_1]\!] = [\![e_2]\!]$.*

We can now prove that property (3) holds.

Proposition 6.2 *For all $e \in \mathbf{P}_c$, $[\![\delta(e)]\!] = [1][\![e]\!]$.*

Proof: By induction on the structure of $e \in \mathbf{P}_c$. By the above discussion, we may assume that e is in canonical form with respect to causes.

- $e = (K \Rightarrow a.p)$. We distinguish two cases depending on whether $K = \emptyset$ or not.

  (1) Assume that $K = \emptyset$. Then

$$\begin{array}{rcll} [\![\delta(e)]\!] & = & [\![(\emptyset \Rightarrow a.p)]\!] & \\ & = & (a, \emptyset).\langle 1 \rangle [\![p]\!] & \\ & = & (a, \emptyset).[2]\langle 1 \rangle [\![p]\!] & \text{by Proposition 6.1 and the 1-boundedness of } [\![p]\!] \\ & = & (a, [1]\emptyset).[2]\langle 1 \rangle [\![p]\!] & \text{as } [1]\emptyset = \emptyset \\ & = & [1]((a, \emptyset).\langle 1 \rangle [\![p]\!]) & \\ & = & [1][\![e]\!]. & \end{array}$$

  (2) Assume that $K = \{n\}$. The proof proceeds as in the above case using Fact 6.1(1), Proposition 6.1 and the 1-boundedness of $[\![p]\!]$. The details are omitted.

- $e = e_1 \odot e_2$, $\odot \in \{+, \between, |\}$. Then

$$\begin{array}{rcll} [\![\delta(e_1 \odot e_2)]\!] & = & [\![\delta(e_1) \odot \delta(e_2)]\!] & \\ & = & [\![\delta(e_1)]\!] \odot [\![\delta(e_2)]\!] & \\ & = & [1][\![e_1]\!] \odot [1][\![e_2]\!] & \text{by induction} \\ & = & [1]([\![e_1]\!] \odot [\![e_2]\!]) & \text{by Lemma 6.1} \\ & = & [1][\![e]\!]. & \square \end{array}$$

We now have enough technical material to relate operational and tree transitions for a term $e \in \mathbf{P}_c$.

Lemma 6.4 *For all $e \in \mathbf{P}_c$, $l \in L$, the following statements hold:*

(1) *$e \stackrel{l}{\longrightarrow}_c e'$ implies $[\![e]\!] \stackrel{l}{\twoheadrightarrow} [\![e']\!]$, and*

(2) *$[\![e]\!] \stackrel{l}{\twoheadrightarrow} x$ implies that $e \stackrel{l}{\longrightarrow}_c e'$ for some e' such that $[\![e']\!] = x$.*

Proof: Statement (1) is proven by induction on the length of the proof of the derivation $e \stackrel{l}{\longrightarrow}_c e'$. Statement (2) is shown by induction on the structure of (the canonical form of) e. The above proposition is used in the proof of both statements. The details are standard and omitted. □

We can now use the agreement between operational and tree-transitions to prove the main result of this section.

Theorem 6.1 *For all $e_1, e_2 \in \mathbf{P}_c$, $e_1 \sim_c e_2$ iff $[\![e_1]\!] = [\![e_2]\!]$.*

Proof: We prove the two implications separately.

(If) It is sufficient to show that the relation

$$R = \{(e_1, e_2) \mid e_1, e_2 \in \mathbf{P}_c \text{ and } [\![e_1]\!] = [\![e_2]\!]\}$$

is a causal bisimulation. This is easily checked using Lemma 6.4.

(Only if) We shall prove that $e_1 \sim_c e_2$ implies $[\![e_1]\!] = [\![e_2]\!]$ by induction on the combined size of e_1 and e_2, $\| e_1 \| + \| e_2 \|$.

- Base case, $\| e_1 \| + \| e_2 \| = 0$. In this case, by Lemma 6.4(2), it is easy to see that $[\![e_1]\!] = [\![e_2]\!] = 0$ as the empty tree is the only one which can not perform a tree transition.
- Inductive step, $\| e_1 \| + \| e_2 \| > 0$. Assume that $e_1 \sim_c e_2$. In order to show that $[\![e_1]\!] = [\![e_2]\!]$, it is sufficient to prove that, for each $l \in L$, $T \in \mathsf{SCT}$,

$$[\![e_1]\!] \stackrel{l}{\twoheadrightarrow} T \quad \text{implies} \quad [\![e_2]\!] \stackrel{l}{\twoheadrightarrow} T. \tag{4}$$

  The claim will then follow by symmetry. Assume then that $[\![e_1]\!] \stackrel{l}{\twoheadrightarrow} T$. By Lemma 6.4(2), there exists e_1' such that $e_1 \stackrel{l}{\longrightarrow}_c e_1'$ and $[\![e_1']\!] = T$. As $e_1 \sim_c e_2$, there exists e_2' such that $e_2 \stackrel{l}{\longrightarrow}_c e_2'$ and $e_1' \sim_c e_2'$. By the inductive hypothesis, $e_1' \sim_c e_2'$ implies $[\![e_1']\!] = [\![e_2']\!] = T$. Moreover, by Lemma 6.4(1), $e_2 \stackrel{l}{\longrightarrow}_c e_2'$ implies $[\![e_2]\!] \stackrel{l}{\twoheadrightarrow} [\![e_2']\!] = T$. This completes the proof of (4).

The proof of the theorem is now complete. □

7 The Completeness Theorem for $\sim_c$

As mentioned above, the proof of the coincidence of distributed, timed and causal equivalence over the language $\mathbf{P}_c$ will be algebraic in nature and will exploit the equational characterizations for $\sim_d$ and $\sim_t$. In this section, we shall prove that an identical axiomatization can be given for causal equivalence over $\mathbf{P}$. The coincidence of distributed, timed and causal equivalence will be an immediate corollary of this fact.

The following proposition collects some basic properties of $\sim_c$ which will find considerable application in what follows and are immediate consequences of Theorem 6.1.

Proposition 7.1 (Some Properties of $\sim_c$)

(1) *Causal equivalence, $\sim_c$, is a congruence over $\mathbf{P}$ and $\mathbf{P}_c$.*

(2) *For all $p, q \in \mathbf{P}$, $n, m \in I\!N_+$, $(\{n\} \Rightarrow p) \sim_c (\{m\} \Rightarrow q)$ implies $p \sim_c q$. Moreover, if $\| (\{n\} \Rightarrow p) \| = \| (\{m\} \Rightarrow q) \| > 0$ then $m = n$.*

For the sake of clarity, let us recall that $=_E$ denotes the least Σ-congruence which satisfies the equations in Figure 3. The first step of the proof of the equational characterization for $\sim_c$ is to show that the equational proof system is sound with respect to it.

Proposition 7.2 (Soundness) *For all $p, q \in \mathbf{P}$, $p =_E q$ implies $p \sim_c q$.*

Proof: In view of Proposition 7.1, it is sufficient to show that each axiom in E is sound with respect to $\sim_c$. The straightforward verification is left to the reader. □

We are thus left to prove that the set of axioms E is indeed complete with respect to $\sim_c$ over $\mathbf{P}$. As usual, the proof of completeness relies on the isolation of suitable normal forms for processes. From § 4 we get for free both a notion of normal form and a normalization result for $\mathbf{P}$ modulo $\sim_c$. For the sake of clarity, we recall that normal forms have the form

$$\sum_{i \in I} a_i.p_i \between q_i,$$

where I is a finite index set and p_i, q_i are themselves normal forms for all $i \in I$. In view of Lemma 4.1 and of the soundness of $=_E$ with respect to $\sim_c$, the completeness of the set of equations E with respect to $\sim_c$ can be reduced to completeness of E over the set of normal forms. In order to prove that this is indeed the case, we shall need a few technical results.

The next lemma gives a general analysis of the initial moves performed by terms in $\mathbf{P}_c$ of the form $(\varnothing \Rightarrow p)$, that are the representation in $\mathbf{P}_c$ of the processes in $\mathbf{P}$, and $(\{n\} \Rightarrow p)$.

Lemma 7.1 *Let $p \in \mathbf{P}$ and $l \in L$. Then:*

(1) *$(\varnothing \Rightarrow p) \stackrel{l}{\longrightarrow}_c x$ implies $l = (a, \varnothing)$, for some $a \in A$, and $x \sim_c (\{1\} \Rightarrow y) | (\varnothing \Rightarrow z)$ for some $y, z \in \mathbf{P}$;*

(2) *$(\{n\} \Rightarrow p) \stackrel{l}{\longrightarrow}_c x$ implies $l = (a, \{n\})$, for some $a \in A$, and $x \sim_c (\{1\} \Rightarrow y) | (\{n+1\} \Rightarrow z)$ for some $y, z \in \mathbf{P}$.*

Proof: Both statements can be shown by induction on the length of the proof of the relevant derivation. The straightforward details are omitted. □

A key result in the proofs of the completeness theorems presented in [CH89] and [AH89] is the so-called "simplification lemma". Not surprisingly, we shall also find it useful to have such a result for $\sim_c$ over $\mathbf{P}_c$. Unfortunately, due to the nature of the causal operational semantics, it seems unlikely that the standard techniques developed in the literature for proving such a result, see e.g. the above given references and [Mol89], can be readily applied in this framework. However, we shall be able to prove a restricted form of the simplification result which will turn out to be sufficient for our purposes.

Notation 7.1 *For each index set $I = \{i_1, \ldots, i_k\}$, $k \geq 0$, the notation $\prod_{i \in I} e_i$ stands for $e_{i_1} | \cdots | e_{i_k}$ and is justified by commutativity and associativity of $|$ modulo $\sim_c$. By convention, $\prod_{i \in \varnothing} e_i = (\varnothing \Rightarrow NIL)$.*

Lemma 7.2 (Simplification) *Let* $\{n_i \mid i \in I\} \subseteq I\!N_+$ *and* $\{p_i \mid i \in I\} \subseteq \mathbf{P}$, *where* I *is a finite index set. Let* $p, q \in \mathbf{P}$ *and* $x = \prod_{i \in I}(\{n_i\} \Rightarrow p_i)$. *Then:*

(1) *for each* $n, m \notin \{n_i \mid i \in I\}$, $(\{n\} \Rightarrow p)|x \sim_c (\{m\} \Rightarrow q)|x$ *implies* $(\{n\} \Rightarrow p) \sim_c (\{m\} \Rightarrow q)$;

(2) $(\varnothing \Rightarrow p)|x \sim_c (\varnothing \Rightarrow q)|x$ *implies* $(\varnothing \Rightarrow p) \sim_c (\varnothing \Rightarrow q)$.

Proof: See Appendix B. □

Our next aim will be to show that, for all $p, q, r, t \in \mathbf{P}$,

$$(\{1\} \Rightarrow p)|(\varnothing \Rightarrow r) \sim_c (\{1\} \Rightarrow q)|(\varnothing \Rightarrow t) \quad \text{implies} \quad p \sim_c q \text{ and } r \sim_c t. \tag{5}$$

This decomposition result will be used in the proof of the completeness theorem and will allow us to prove such a theorem by means of some kind of induction. Note that all the terms in $\mathbf{P}_c$ reachable from terms of the form $(\varnothing \Rightarrow \sum_{i \in I} a_i.p_i \not| q_i)$, the representation of normal forms in $\mathbf{P}_c$, by means of one step transitions are of the form $(\{1\} \Rightarrow p)|(\varnothing \Rightarrow r)$, for some $p, r \in \mathbf{P}$.

Statement (5) will follow in a straightforward way from a rather general decomposition result for parallel processes. The following notation will be useful in the proof of such a result.

Notation 7.2 *Let* $\{(\{n_i\} \Rightarrow p_i) \mid i \in I\}$ *and* $\{(\{m_j\} \Rightarrow q_j) \mid j \in J\}$ *be two indexed sets of terms. We write* $\{(\{n_i\} \Rightarrow p_i) \mid i \in I\} \sim_c \{(\{m_j\} \Rightarrow q_j) \mid j \in J\}$ *iff there exists a bijection* $\phi : I \to J$ *such that, for all* $i \in I$,

(a) $n_i = m_{\phi(i)}$ *and*

(b) $p_i \sim_c q_{\phi(i)}$.

Theorem 7.1 (Parallel Decomposition) *Let* $\{n_i \mid i \in I\} \subseteq I\!N_+$, $\{m_j \mid j \in J\} \subseteq I\!N_+$, $\{p_i \mid i \in I\} \subseteq \mathbf{P}$ *and* $\{q_j \mid j \in J\} \subseteq \mathbf{P}$ *be such that*

(1) *for all* $i_1, i_2 \in I$, $i_1 \neq i_2$ *implies* $n_{i_1} \neq n_{i_2}$,

(2) *for all* $j_1, j_2 \in J$, $j_1 \neq j_2$ *implies* $m_{j_1} \neq m_{j_2}$, *and*

(3) *for all* $i \in I$, $j \in J$, $\| (\{n_i\} \Rightarrow p_i) \| > 0$ *and* $\| (\{m_j\} \Rightarrow q_j) \| > 0$.

Let $r, t \in \mathbf{P}$. *Then* $(\prod_{i \in I}(\{n_i\} \Rightarrow p_i))|(\varnothing \Rightarrow r) \sim_c (\prod_{j \in J}(\{m_j\} \Rightarrow q_j))|(\varnothing \Rightarrow t)$ *implies* $\{(\{n_i\} \Rightarrow p_i) \mid i \in I\} \sim_c \{(\{m_j\} \Rightarrow q_j) \mid j \in J\}$ *and* $r \sim_c t$.

Proof: See Appendix C. □

Using this theorem we may now prove statement (5) announced before.

Corollary 7.1 *Let* $p, q, r, t \in \mathbf{P}$. *Assume that* $(\{1\} \Rightarrow p)|(\varnothing \Rightarrow r) \sim_c (\{1\} \Rightarrow q)|(\varnothing \Rightarrow t)$. *Then* $p \sim_c q$ *and* $r \sim_c t$.

Proof: Assume that $(\{1\} \Rightarrow p)|(\varnothing \Rightarrow r) \sim_c (\{1\} \Rightarrow q)|(\varnothing \Rightarrow t)$. First of all, note that, by Lemma 7.1, $p \sim_c NIL$ iff $q \sim_c NIL$. Similarly, $r \sim_c NIL$ iff $t \sim_c NIL$.

If $p \sim_c NIL$, i.e. $\| (\{1\} \Rightarrow p) \| = 0$, then $q \sim_c NIL$ by the above observation. Moreover, by Fact 5.1(3), $(\varnothing \Rightarrow r) \sim_c (\varnothing \Rightarrow t)$, i.e. $r \sim_c t$.

If $r \sim_c NIL$ then $t \sim_c NIL$ by the above observation. Moreover, by Fact 5.1(3), $(\{1\} \Rightarrow p) \sim_c (\{1\} \Rightarrow q)$. By Proposition 7.1(3), this implies that $p \sim_c q$.

Assume now that $\| (\{1\} \Rightarrow p) \| > 0$ and $\| (\varnothing \Rightarrow r) \| > 0$. Then, by symmetry, $\| (\{1\} \Rightarrow q) \| > 0$. We may then apply the Parallel Decomposition Theorem to obtain that $p \sim_c q$ and $r \sim_c t$. □

We have now developed all the technical material which will be needed to prove the promised completeness theorem for causal equivalence.

Theorem 7.2 (Completeness) *For each $p, q \in \mathbf{P}$, $p \sim_c q$ implies $p =_E q$*

Proof: Assume that $p \sim_c q$. We prove the claim by induction on $\| p \| + \| q \|$. By Lemmas 4.1 and 7.2, we may assume that $p = \sum_{i \in I} a_i.p_i \between p'_i$ and $q = \sum_{j \in J} b_j.q_j \between q'_j$ are normal forms. Assume then that $p = \sum_{i \in I} a_i.p_i \between p'_i \sim_c \sum_{j \in J} b_j.q_j \between q'_j = q$. We shall show that $p =_E q$ by proving that $p =_E p + q =_E q$. By symmetry, it is sufficient to establish that, for each $j \in J$,

$$p =_E p + b_j.q_j \between q'_j.$$

Let $j \in J$. Then, by the causal operational semantics,

$$(\varnothing \Rightarrow q) \stackrel{(b_j,\varnothing)}{\longrightarrow}_c (\{1\} \Rightarrow q_j) | (\varnothing \Rightarrow q'_j).$$

As $p \sim_c q$, i.e. $(\varnothing \Rightarrow p) \sim_c (\varnothing \Rightarrow q)$, there exists $i \in I$ such that $a_i = b_j$, $(\varnothing \Rightarrow p) \stackrel{(a_i,\varnothing)}{\longrightarrow}_c (\{1\} \Rightarrow p_i) | (\varnothing \Rightarrow p'_i)$ and

$$(\{1\} \Rightarrow p_i) | (\varnothing \Rightarrow p'_i) \sim_c (\{1\} \Rightarrow q_j) | (\varnothing \Rightarrow q'_j).$$

By Corollary 7.1, this implies that $p_i \sim_c q_j$ and $p'_i \sim_c q'_j$. As $\| p_i \|, \| p'_i \| < \| p \|$ and $\| q_j \|, \| q'_j \| < \| q \|$, we may apply the inductive hypothesis to obtain that $p_i =_E q_j$ and $p'_i =_E q'_j$. By substitutivity,

$$a_i.p_i \between p'_i =_E a_i.q_j \between q'_j = b_j.q_j \between q'_j.$$

Hence, by (A1)-(A3), $p =_E p + a_i.p_i \between p'_i =_E p + b_j.q_j \between q'_j$. This implies that $p =_E p + q$. The claim now follows by symmetry. □

As a corollary of the completeness theorem for $\sim_c$ over $\mathbf{P}$, we can now prove that causal, distributed and timed equivalence coincide over the language considered in this paper. Hence the notions of observability underlying these equivalences, although based on very different intuitions, have equal discriminating power over $\mathbf{P}$.

Theorem 7.3 *For all $p, q \in \mathbf{P}$, $p \sim_c q$ iff $p \sim_d q$ iff $p \sim_t q$.*

Indeed, Theorem 7.2 may be strengthened by showing that the set of equations E is *ω-complete*, i.e. complete for the open term theory, with respect to $\sim_c$, and, a fortiori, also for $\sim_d$ and $\sim_t$. Let Var be a countable set of *variables* ranged over by x, y, z. $\mathbb{P}$ $(t, t' \in \mathbb{P})$ will denote the set of expressions built by adding the clause

- $x \in \mathsf{Var}$ implies $x \in \mathbb{P}$

to the formation rules for $\mathbf{P}$. The equivalence $\sim_c$ can now be extended to $\mathbb{P}$ in the standard way as follows:

Definition 7.1 *Let $t, t' \in \mathbb{P}$. Then $t \sim_c t'$ iff for all closed substitutions $\sigma : \mathsf{Var} \to \mathbf{P}$, $t\sigma \sim_c t'\sigma$. An equational theory EQ over the signature of $\mathbf{P}$ is then called* ω-complete *with respect to $\sim_c$ iff for all open terms $t, t' \in \mathbb{P}$, $t \sim_c t'$ iff $EQ \vdash t = t'$. $EQ \vdash t = t'$ will also be written as $t =_{EQ} t'$.*

We shall now prove that the set of axioms E presented in Figure 3 is indeed ω-complete with respect to $\sim_c$ over $\mathbb{P}$. In the proof we shall make use of a technique for proving the ω-completeness of a set of equations developed by J.F. Groote in [Gro90]. For the sake of clarity, we shall now briefly outline Groote's proof-technique for showing the ω-completeness of a set of equations. Assume that t and t' are open terms in $\mathbb{P}$ and $t \sim_c t'$, i.e., by Theorem 7.2, $t\sigma =_E t'\sigma$ for all closed substitutions σ. The application of Groote's technique requires the isolation of a closed substitution $\rho : \mathsf{Var} \to \mathbf{P}$, mapping each variable occurring in t and t' to a distinguished closed term representing this variable, and of a translation map $R : \mathbf{P} \to \mathbb{P}$, which replaces each subterm representing a variable by the variable itself. This pair of functions is required to satisfy the following conditions:

(1) $t =_E R(\rho(t))$ and $t' =_E R(\rho(t'))$,

(2) for each $f \in \Sigma - \{NIL\}$ of arity n and $p_1, \ldots, p_n, q_1, \ldots, q_n \in \mathbf{P}$, $R(f(p_1, \ldots, p_n)) =_{E'} R(f(q_1 \ldots q_n))$, where $E' = E \cup \{R(p_i) = R(q_i) \mid i = 1, \ldots, n\}$, and

(3) for each axiom $t_1 = t_2$ in E and closed substitution σ, $R(\sigma(t_1)) =_E R(\sigma(t_2))$.

Having found such a pair of maps ρ and R satisfying conditions (1)-(3) above, we could then obtain the ω-completeness of E with respect to $\mathbb{P}$ by applying the following instance of Theorem 3.1 from [Gro90], page 317.

Theorem 7.4 *If for each $t, t' \in \mathbb{P}$ such that $t\sigma =_E t'\sigma$, for all closed substitutions σ, there exist a closed substitution $\rho : \mathsf{Var} \to \mathbf{P}$ and a map $R : \mathbf{P} \to \mathbb{P}$ satisfying (1)-(3) above then E is ω-complete.*

We shall now apply the technique described above to prove that E is indeed ω-complete with respect to $\sim_c$ over $\mathbb{P}$.

Proposition 7.3 (ω-Completeness) *For all $t, t' \in \mathbb{P}$, $t \sim_c t'$ iff $t =_E t'$.*

Proof: Let $t, t' \in \mathbb{P}$ be such that $t \sim_c t'$. By Theorem 7.4, in order to prove that E is ω-complete, it is sufficient to find $\rho : \mathsf{Var} \to \mathbf{P}$ and $R : \mathbf{P} \to \mathbb{P}$ satisfying conditions (1)-(3) above. Define $\rho : \mathsf{Var} \to \mathbf{P}$ by $\rho(x) = a_x.NIL$, where a_x is a distinguished action for each $x \in \mathsf{Var}$ that does not occur in t or t'. (Note that we rely upon the fact that A is countable.) The translation map $R : \mathbf{P} \to \mathbb{P}$ is defined by induction on the structure of p as follows:

- $R(nil) = nil$,
- $R(a.p) = \begin{cases} x/R(p) & \text{if } a = a_x \\ a.R(p) & \text{otherwise,} \end{cases}$
- $R(p \odot q) = R(p) \odot R(q)$, for $\odot \in \Sigma_2$.

The proof of the fact that ρ and R satisfy conditions (1)-(3) mimicks the one of Theorem 5.1.1 in [Gro90] and is thus omitted. □

7.1 Full Abstraction for the Causal Tree Model

To end this paper, we shall now show how all the equivalences we have studied for the language **P** can be obtained by applying Abramsky's testing scenario for bisimulation equivalence [Abr87] in contexts built using a simple variation on the operator of *action refinement* [AH89, NEL89]. This shows that, at least for the simple language we have considered in this paper, the observation of the causal structure of a process, of its distribution in space and of actions with duration can all be reduced to the application "in context" of a standard interleaving testing scenario. For the language **P**, a simple notion of syntactic action refinement suffices for this purpose.

Definition 7.2 (Action Refinement) *An* action refinement *ρ is a map $\rho : A \to A^+$. For each $p \in \mathbf{P}$ and action refinement ρ, $p\rho$ denotes the process (in **P**) given by:*

- $NIL\rho = NIL$,
- $(a.p)\rho = a_1.\ldots.a_n.(p\rho)$, *if* $\rho(a) = a_1.\ldots.a_n$, *and*
- $(p \odot q)\rho = p\rho \odot q\rho$, *for all* $\odot \in \Sigma_2$.

The closure of standard bisimulation with respect to all refinements, denoted by $\sim^\rho$, is given by

$$\text{for all } p,q \in \mathbf{P},\ p \sim^\rho q \text{ iff for all action refinements } \rho,\ p\rho \sim q\rho.$$

As it is well-known, $\sim^\rho$ is the largest congruence over $\mathbf{P}$ contained in $\sim$ which is preserved by action refinements. Intuitively, it can be seen as arising by applying Abramsky's testing scenario for bisimulation equivalence in all refinement contexts. We shall now show that $\sim^\rho$ coincides with $\sim_c$, and thus also with distributed and timed bisimulation equivalences, over $\mathbf{P}$.

Theorem 7.5 *For all $p,q \in \mathbf{P}$, $p \sim^\rho q$ iff $p \sim_c q$.*

Proof: (Sketch) We give an outline of the proof of the two implications. The details follow those of similar proofs in [AH89] and are omitted.

(Only if) By Theorem 7.3, it is sufficient to prove that $p \sim^\rho q$ implies $p \sim_t q$. Let ρ_t denote the action refinement given by $\rho_t(a) = S(a).F(a)$, for all $a \in A$. Then it is not too difficult to prove that, for all $p,q \in \mathbf{P}$, $p \sim_t q$ iff $p\rho_t \sim q\rho_t$. The claim then follows because $p \sim^\rho q$ implies $p\rho_t \sim q\rho_t$.

(If) As $\sim$ is a Σ-congruence, so is $\sim^\rho$. In order to prove that $p \sim_c q$ implies $p \sim^\rho q$, it is thus sufficient to prove that $\sim^\rho$ satisfies all the equations that completely characterize $\sim_c$. This follows easily by observing that ρ distributes over all the binary operators in Σ and that $\sim$ satisfies all of the equations in Figure 3. □

As a corollary of the above result, we obtain the full abstractness of the causal trees semantics for the language $\mathbf{P}$ with respect to the behavioural equivalence $\sim^\rho$.

Theorem 7.6 (Full Abstraction) *For all $p,q \in \mathbf{P}$, $p \sim^\rho q$ iff $[\![p]\!] = [\![q]\!]$.*

8 Conclusions

In this paper, we proved that three well-known noninterleaving equivalences, namely *timed equivalence* [Hen88], *distributed bisimulation* [CH89] and *causal equivalence* [DD88], coincide over a simple language for processes without communication and restriction. This result shows that, maybe not surprisingly, the explicit observation of the causal links among events, of the distributed structure of processes and of the duration of action occurrences have the same discriminating power over such a simple language. Although the language under consideration is rather poor, we believe that Theorem 7.3 has some interest because of the intrinsically different nature of the observations underlying the three equivalences considered in this paper. Indeed, Theorem 7.3 is highly language-dependent and does not hold for richer languages and the versions of the equivalences under consideration which abstract from the internal evolution of processes. In this richer world, the three observational scenarios have incomparable discriminating power. The interested reader is invited to consult [BCHK91] for a good discussion of examples showing that two distribution based equivalences, which evolved from distributed bisimulation, are incomparable with causal bisimulation. Moreover, a weak version of timed equivalence [Hen88] is incomparable with the weak distributed bisimulations proposed in [CH89, BCHK91] and weak causal bisimulation [DD88]. In fact, the latter equivalences satisfy the equation

$$a.(x + \tau.y) \quad = \quad a.(x + \tau.y) + a.y$$

while weak timed equivalence does not.

The equivalences considered in this paper are, in the author's opinion, representative of the three different observational scenarios mentioned above. However, Theorem 7.3 can be extended to several other notions of equivalence. For instance, it can be shown that also *ST-bisimulation* [GV87] (in its reformulation as *refine equivalence* [AH89]) and the *location equivalence* of [BCHK91], both in its more refined and loose forms, coincide with $\sim_c$ over the language we have considered. On the other hand, it follows from an example presented in [BC88] and due to Matthew Hennessy that *pomset bisimulation* is strictly coarser than all the afore-mentioned equivalences even over the simple processes under consideration.

As a consequence of the completeness result for causal bisimulation (Theorem 7.2), we also showed that the causal trees model for the language **P** is fully abstract with respect to the largest **P**-congruence preserved by action-refinement and contained in strong bisimulation equivalence. Thus, for this simple language, one can justify the observation of causality, in its strongest form, on the basis of sequential observations and action refinement. However, the problem of finding (preferably) natural operators which allow one to justify causal bisimulation over a language with synchronization and restriction in terms of some form of sequential observation remains open. In the author's opinion, this is a very interesting and important subject for further research.

In this paper, the comparison between different semantics for the simple language under consideration has been carried out using algebraic, axiomatic techniques. This was made possible by the existence of equational axiomatizations for causal, distributed and timed bisimulation which are very abstract in nature. Technically, the abstract nature of the equational characterizations presented in this paper is reflected in the fact that they are ω-complete. Unfortunately, to the best of the author's knowledge, no ω-complete axiomatization of non-trivial behavioural equivalences is known for languages which include CCS restriction or any other encapsulation/hiding operators. (See [Mol89, Gro90] for some examples of ω-complete axiomatizations of behavioural equivalences over several restriction-free languages.) All the axiomatizations of non-interleaving behavioural equivalences for rich, finite process algebras presented in the literature rely on some form of expansion theorem à là Milner [Mil80] and are thus very operational in nature. As a consequence, the author has found them to be of limited use in pin-pointing the differences and the relationships between different non-interleaving semantics. This is essentially due to the complexity of the auxiliary prefixing operations used in deriving complete axiomatizations for non-interleaving equivalences in, e.g., [DD88, BCHK91, FGM91]. In these axiomatizations virtually all the information about the semantics is hidden inside the auxiliary prefixing operations and very little insight can be gained from the equations themselves, which are, by and large, those presented by Hennessy and Milner's in [HM85]. The search for ω-complete axiomatizations of behavioural equivalences for finite process algebras is, at least in the author's opinion, a very important topic for future reasearch.

A promising approach to the development of more abstract proof systems for non-interleaving equivalences is presented in [Hen91]. In that paper, a proof system for ST-bisimulation equivalence, which does not use an expansion theorem, is developed. The proof system is based upon a standard set of equational laws and on a set of inference rules to deal with the restriction combinator. The interested reader is invited to consult the afore-mentioned reference for some examples of the power of this approach. Unfortunately, however, a certain amount of operational information needs to be built into one of the rules in [Hen91] (see page 27 of that reference) to obtain a complete inference system.

Acknowledgements: The work reported in this paper owes a great deal to the seminal work and influence of Ilaria Castellani, Philippe Darondeau, Pierpaolo Degano and Matthew Hennessy. Many thanks to Gérard Boudol for his encouragement to write this little paper.

References

[Abr87] S. Abramsky. Observation equivalence as a testing equivalence. *Theoretical Computer Science*, 53:225–241, 1987.

[Ace91] L. Aceto. Full abstraction for series-parallel pomsets. In S. Abramsky and T. Maibaum, editors, *Proceedings TAPSOFT '91*, volume 493 of *Lecture Notes in Computer Science*, pages 1–25. Springer-Verlag, 1991. A full version appeared as Computer Science Technical Report 2/90, University of Sussex, March 1990.

[AH89] L. Aceto and M. Hennessy. Towards action-refinement in process algebras. In *Proceedings* 4^{th} *Annual Symposium on Logic in Computer Science,* Asilomar, California, pages 138–145. IEEE Computer Society Press, 1989. Full version to appear in *Information and Computation.*

[BC88] G. Boudol and I. Castellani. Concurrency and atomicity. *Theoretical Computer Science*, 59(1/2):25–84, 1988.

[BCHK91] G. Boudol, I. Castellani, M. Hennessy, and A. Kiehn. A theory of processes with localities. Technical Report 13/91, University of Sussex, December 1991.

[BK84] J.A. Bergstra and J.W. Klop. Process algebra for synchronous communication. *Information and Computation*, 60(1/3):109–137, 1984.

[BW90] J.C.M. Baeten and W.P. Weijland. *Process Algebra.* Cambridge Tracts in Theoretical Computer Science 18. Cambridge University Press, 1990.

[Cas88] I. Castellani. *Bisimulations for concurrency.* PhD thesis, Report CST-51-88, Department of Computer Science, University of Edinburgh, April 1988.

[CH89] I. Castellani and M. Hennessy. Distributed bisimulations. *Journal of the ACM*, 36(4):887–911, 1989.

[DD88] P. Darondeau and P. Degano. Causal trees. Publication Interne No. 442, IRISA, Rennes Cedex, France, 1988. An extended abstract appeared in: Proceedings ICALP 89, Stresa (G. Ausiello, M. Dezani-Ciancaglini & S. Ronchi Della Rocca), LNCS 372, Springer-Verlag, pp. 234–248.

[DD90] P. Darondeau and P. Degano. Causal trees: interleaving + causality. In I. Guessarian, editor, *Semantics of Systems of Concurrent Processes, Proceedings LITP Spring School on Theoretical Computer Science,* La Roche Posay, France, volume 469 of *Lecture Notes in Computer Science*, pages 239–255. Springer-Verlag, 1990.

[DDNM87] P. Degano, R. De Nicola, and U. Montanari. Observational equivalences for concurrency models. In M. Wirsing, editor, *Formal Description of Programming Concepts – III, Proceedings of the* 3^{th} *IFIP WG 2.2 working conference,* Ebberup 1986, pages 105–129. North-Holland, 1987.

[FGM91] G. Ferrari, R. Gorrieri, and U. Montanari. An extended expansion theorem. In S. Abramsky and T. Maibaum, editors, *Proceedings TAPSOFT '91*, volume 494 of *Lecture Notes in Computer Science*, pages 29–48. Springer-Verlag, 1991.

[GG89] R.J. van Glabbeek and U. Goltz. Equivalence notions for concurrent systems and refinement of actions. Arbeitspapiere der GMD 366, Gesellschaft für Mathematik und Datenverarbeitung, Sankt Augustin, 1989. An extended abstract appeared in: Proceedings 14^{th} Symposium on Mathematical Foundations of Computer Science, Porąbka-Kozubnik, Poland, August/September 1989 (A. Kreczmar and G. Mirkowska), LNCS 379, Springer-Verlag, pp. 237–248.

[Gro90] J.F. Groote. A new strategy for proving ω–completeness with applications in process algebra. In J.C.M. Baeten and J.W. Klop, editors, *Proceedings CONCUR 90,* Amsterdam, volume 458 of *Lecture Notes in Computer Science*, pages 314–331. Springer-Verlag, 1990.

[GV87] R.J. van Glabbeek and F.W. Vaandrager. Petri net models for algebraic theories of concurrency. In J.W. de Bakker, A.J. Nijman, and P.C. Treleaven, editors, *Proceedings PARLE conference,* Eindhoven*, Vol. II (Parallel Languages)*, volume 259 of *Lecture Notes in Computer Science*, pages 224–242. Springer-Verlag, 1987.

[Hen88] M. Hennessy. Axiomatising finite concurrent processes. *SIAM Journal on Computing*, 17(5):997–1017, 1988.

[Hen91] M. Hennessy. A proof system for weak ST-bisimulation over a finite process algebra. Computer Science Report 6/91, University of Sussex, June 1991.

[HM85] M. Hennessy and R. Milner. Algebraic laws for nondeterminism and concurrency. *Journal of the ACM*, 32(1):137–161, 1985.

[Mil80] R. Milner. *A Calculus of Communicating Systems*, volume 92 of *Lecture Notes in Computer Science*. Springer-Verlag, 1980.

[Mil89] R. Milner. *Communication and Concurrency.* Prentice-Hall International, Englewood Cliffs, 1989.

[Mol89] F. Moller. *Axioms for concurrency.* PhD thesis, Report CST-59-89, Department of Computer Science, University of Edinburgh, 1989.

[NEL89] M. Nielsen, U. Engberg, and K.**S**. Larsen. Fully abstract models for a process language with refinement. In J.W. de Bakker, W.-P. de Roever, and G. Rozenberg, editors, *REX School/Workshop on Linear Time, Branching Time and Partial Order in Logics and Models for Concurrency,* Noordwijkerhout, volume 354 of *Lecture Notes in Computer Science*, pages 523–548. Springer-Verlag, 1989.

[Par81] D.M.R. Park. Concurrency and automata on infinite sequences. In P. Deussen, editor, 5^{th} *GI Conference*, volume 104 of *Lecture Notes in Computer Science*, pages 167–183. Springer-Verlag, 1981.

[Plo81] G.D. Plotkin. A structural approach to operational semantics. Report DAIMI FN-19, Computer Science Department, Aarhus University, 1981.

[RT88] A. Rabinovich and B.A. Trakhtenbrot. Behavior structures and nets. *Fundamenta Informaticae*, 11(4):357–404, 1988.

[Vaa90] F.W. Vaandrager. *Algebraic Techniques for Concurrency and Their Application.* PhD thesis, University of Amsterdam, 1990.

Appendix A: Proof of Results in Section 6

This appendix contains the proofs of some of the results stated without proof in § 6.

Proof of Proposition 6.1: By induction on the depth of the n-bounded tree $T = \sum_{i \in I}(a_i, K_i).T_i$. By the definition of the auxiliary operators,

$$[n+1](\langle n \rangle T) = \sum_{i \in I}(a_i, [n+1]\langle n \rangle K_i).[n+2]\langle n+1 \rangle T_i.$$

As T is n-bounded, T_i is $n+1$-bounded for each $i \in I$. It is then possible to apply the inductive hypothesis to $[n+2]\langle n+1 \rangle T_i$ to obtain that, for each $i \in I$,

$$[n+2]\langle n+1 \rangle T_i \quad = \quad \langle n+1 \rangle T_i. \tag{6}$$

We shall now prove that, as T is n-bounded, for each $i \in I$

$$[n+1]\langle n \rangle K_i \quad = \quad \langle n \rangle K_i. \tag{7}$$

In order to prove (7) we distinguish two subcases depending on whether $K_i = \varnothing$ or $K_i = \{m\} \subseteq I\!N_+$.

- If $K_i = \varnothing$ then $[n+1]\langle n \rangle \varnothing = [n+1]\{n\} = \{n\} = \langle n \rangle \varnothing$.
- If $K_i = \{m\}$ then, as T is n-bounded, $m \leq n$. Thus

$$[n+1]\langle n \rangle \{m\} = [n+1]\{m\} = \{m\} = \langle n \rangle \{m\}.$$

Hence we have shown that (7) holds. By putting (6) and (7) together we have that

$$\begin{aligned} [n+1](\langle n \rangle T) &= \sum_{i \in I}(a_i, \langle n \rangle K_i).\langle n+1 \rangle T_i \\ &= \langle n \rangle T. \ \square \end{aligned}$$

Proof of Fact 6.2: Both statements are shown by induction on the depth of $T = \sum_{i \in I}(a_i, K_i).T_i$. We only prove (2). The details of the proof of (1) are similar and left to the reader.

(2) By the definition of the operators $[n]$, we have that

$$[n+m][m]T \quad = \quad \sum_{i \in I}(a_i, [n+m][m]K_i).[n+m+1][m+1]T_i.$$

We may now apply the inductive hypothesis to $[n+m+1][m+1]T_i$ to obtain that $[n+m+1][m+1]T_i = [m+1][n+m]T_i$. We shall now prove that

$$[n+m][m]K_i \quad = \quad [m][(n+m)-1]K_i. \tag{8}$$

The proof of equality (8) proceeds by distinguishing two cases, depending on whether K_i is empty or not. The case $K_i = \varnothing$ is trivially seen to hold as both sides of (8) denote $\varnothing$. We are thus left to examine the case $K_i = \{h\}$, $h \in I\!N_+$. We distinguish the following three sub-cases:

(a) $h < m$,

(b) $n + m \leq h$, or

(c) $m \leq h \leq (n+m) - 1$.

We examine the three possibilities in turn. If (a) holds then it is easy to check by calculation that both sides of (8) evaluate to $\{h\}$. If (b) holds then both sides of equality (8) evaluate to $\{h+2\}$. We are thus left to examine case (c). Assume then that $m \leq h \leq (n+m)-1$. Now

$$\begin{aligned}[n+m][m]\{h\} &= [n+m]\{h+1\} \\ &= \begin{cases} \{h+2\} & \text{if } n+m = h+1 \\ \{h+1\} & \text{if } h+1 < n+m. \end{cases}\end{aligned}$$

By a similar calculation, it is possible to show that

$$[m][(n+m)-1]\{h\} = \begin{cases} \{h+2\} & \text{if } h = (n+m)-1 \\ \{h+1\} & \text{if } h < (n+m)-1. \end{cases}$$

We have thus shown that equality (8) holds. Property (2) now follows immediately. □

Proof of Lemma 6.1: Statement (1) is trivially seen to hold. Statements (2) and (3) are shown by induction on the combined depth of the trees T and U. We only give the proof of statement (2).

(2) Let $T = \sum_{i\in I}(a_i, K_i).T_i$ and $U = \sum_{j\in J}(b_j, H_j).U_j$. By calculation we have that

$$\begin{aligned}[n](T|U) &= [n](\sum_{i\in I}(a_i, K_i).(T_i|[1]U) + \sum_{j\in J}(b_j, H_j).(U_j|[1]T)) \\ &= \sum_{i\in I}(a_i, [n]K_i).[n+1](T_i|[1]U) + \sum_{j\in J}(b_j, [n]H_j).[n+1](U_j|[1]T).\end{aligned}$$

By applying induction and Fact 6.2(2), we then have that

$$\begin{aligned}[n](T|U) &= \textstyle\sum_{i\in I}(a_i, [n]K_i).([n+1]T_i|[n+1][1]U) + \sum_{j\in J}(b_j, [n]H_j).([n+1]U_j|[n+1][1]T) \\ &\quad \text{by induction} \\ &= \textstyle\sum_{i\in I}(a_i, [n]K_i).([n+1]T_i|[1][n]U) + \sum_{j\in J}(b_j, [n]H_j).([n+1]U_j|[1][n]T) \\ &\quad \text{by Fact 6.2(2)} \\ &= (\textstyle\sum_{i\in I}(a_i, [n]K_i).[n+1]T_i)|(\sum_{j\in J}(b_j, [n]H_j).[n+1]U_j) \\ &= [n]T|[n]U.\end{aligned}$$

□

Proof of Lemma 6.2: By induction on the structure of p. We only examine two selected cases of the proof.

- $p = a.q$. By the definition of $[\![\cdot]\!]$, $[\![a.q]\!] = (a, \varnothing).\langle 1\rangle[\![q]\!]$. By the inductive hypothesis, $[\![q]\!]$ is 1-bounded. It is now easy to see that

  Claim: if $T \in \mathsf{SCT}$ is 1-bounded then so is $\langle 1\rangle T$.

  Using the above claim we have that $\langle 1\rangle[\![q]\!]$ is 1-bounded and this implies that $(a, \varnothing).\langle 1\rangle[\![q]\!]$ is also 1-bounded.

- $p = q \between r$. By the definition of $[\![\cdot]\!]$, $[\![q \between r]\!] = [\![q]\!] \between [\![r]\!]$ where, by induction, $[\![q]\!]$ and $[\![r]\!]$ are 1-bounded. Let $[\![q]\!] = \sum_{i\in I}(a_i, K_i).T_i$. Then

  $$[\![q \between r]\!] = \sum_{i\in I}(a_i, K_i).(T_i|[1][\![r]\!]).$$

  We shall now prove that, for each $i \in I$, $T_i|[1][\![r]\!]$ is 2-bounded. The 1-boundedness of $[\![q \between r]\!]$ will follow immediately from this fact.

  The 2-boundedness of $T_i|[1][\![r]\!]$ follows from the following two properties:

(1) if $T \in \mathsf{SCT}$ is n-bounded then $[1]T$ is $n+1$-bounded, and

(2) if $T, U \in \mathsf{SCT}$ are n-bounded then $T|U$ is n-bounded.

Property (1) is easily proven by calculation. Property (2) is shown by induction on the combined depth of T and U. By using (1) and (2), it is easy to see that $T_i|[1][\![r]\!]$ is 2-bounded. In fact, as $[\![q]\!]$ is 1-bounded, T_i is 2-bounded. By (1) and induction, $[1][\![r]\!]$ is also 2-bounded. Hence, by (2), $T_i|[1][\![r]\!]$ is 2-bounded, for each $i \in I$. □

Proof of Lemma 6.3: By induction on the combined depth of T and U. Let $T = \sum_{i\in I}(a_i, K_i).T_i$ and $U = \sum_{j\in J}(b_j, H_j).U_j$. Then we can reason as follows

$$\begin{array}{rcl}
\langle n\rangle(T|U) & = & \langle n\rangle(T \between U + U \between T) \\
& = & \langle n\rangle(\sum_{i\in I}(a_i, K_i).(T_i|[1]U) + \sum_{j\in J}(b_j, H_j).(U_j|[1]T)) \\
& = & \sum_{i\in I}(a_i, \langle n\rangle K_i).\langle n+1\rangle(T_i|[1]U) + \sum_{j\in J}(b_j, \langle n\rangle H_j).\langle n+1\rangle(U_j|[1]T)) \\
& = & \sum_{i\in I}(a_i, \langle n\rangle K_i).(\langle n+1\rangle T_i|\langle n+1\rangle[1]U) + \sum_{j\in J}(b_j, \langle n\rangle H_j).(\langle n+1\rangle U_j|\langle n+1\rangle[1]T) \\
& & \text{by the inductive hypothesis} \\
& = & \sum_{i\in I}(a_i, \langle n\rangle K_i).(\langle n+1\rangle T_i|[1]\langle n\rangle U) + \sum_{j\in J}(b_j, \langle n\rangle H_j).(\langle n+1\rangle U_j|[1]\langle n\rangle T) \\
& & \text{by Fact 6.2(1)} \\
& = & \langle n\rangle T|\langle n\rangle U.
\end{array}$$ □

Appendix B: Proof of the Simplification Lemma

Proof of Lemma 7.2: In order to prove both the statements of the lemma, it is sufficient to show that the relation $\sim_c \circ R \circ \sim_c$ is a causal bisimulation, where R is given by

$$\begin{array}{rrl}
R = \{(e_1, e_2) \mid & (i) & e_1 = (\prod_{h\in H}(\{n_h\} \Rightarrow t_h))|(\varnothing \Rightarrow t) \text{ for some } \{n_h \mid h \in H\} \subseteq I\!N_+ \text{ and} \\
& & \{t_h \mid h \in H\} \subseteq \mathbf{P}, \\
& (ii) & e_2 = (\prod_{k\in K}(\{m_k\} \Rightarrow r_k))|(\varnothing \Rightarrow r) \text{ for some } \{m_k \mid k \in K\} \subseteq I\!N_+ \text{ and} \\
& & \{r_k \mid k \in K\} \subseteq \mathbf{P}, \text{ and} \\
& (iii) & \text{there exists } x = \prod_{i\in I}(\{n_i\} \Rightarrow p_i) \text{ such that } e_1|x \sim_c e_2|x \text{ and} \\
& & (\{n_h \mid h \in H\} \cup \{m_k \mid k \in K\}) \cap \{n_i \mid i \in I\} = \varnothing\}.
\end{array}$$

Assume, in fact, that $\sim_c \circ R \circ \sim_c$ is a causal bisimulation. Then, letting x be as in the proviso of the lemma, we can prove (1) and (2) as follows:

(1) Assume that $n, m \notin \{n_i \mid i \in I\}$ and $(\{n\} \Rightarrow p)|x \sim_c (\{m\} \Rightarrow q)|x$. Then,

$$\begin{array}{rcll}
(\{n\} \Rightarrow p) & \sim_c & (\{n\} \Rightarrow p)|(\varnothing \Rightarrow NIL) & \\
& R & (\{m\} \Rightarrow q)|(\varnothing \Rightarrow NIL) & \text{by the hypotheses of statement (1) of the lemma} \\
& \sim_c & (\{m\} \Rightarrow q). &
\end{array}$$

As $\sim_c \circ R \circ \sim_c$ is a causal bisimulation, this implies that $(\{n\} \Rightarrow p) \sim_c (\{m\} \Rightarrow q)$.

(2) Assume that $(\varnothing \Rightarrow p)|x \sim_c (\varnothing \Rightarrow q)|x$. Then,

$$\begin{array}{rcll}
(\varnothing \Rightarrow p) & \sim_c & (\prod_{h\in\varnothing}(\{n_h\} \Rightarrow t_h))|(\varnothing \Rightarrow p) & \text{as } \prod_{h\in\varnothing}(\{n_h\} \Rightarrow t_h) = (\varnothing \Rightarrow NIL) \\
& R & (\prod_{k\in\varnothing}(\{m_k\} \Rightarrow r_k))|(\varnothing \Rightarrow q) & \text{by the hypothesis of statement (2) of the lemma} \\
& \sim_c & (\varnothing \Rightarrow q). &
\end{array}$$

As $\sim_c \circ R \circ \sim_c$ is a causal bisimulation, this implies that $(\varnothing \Rightarrow p) \sim_c (\varnothing \Rightarrow q)$.

We are thus left to show that $\sim_c \circ R \circ \sim_c$ is a causal bisimulation. By symmetry and the transitivity of $\sim_c$, this will follow if we prove that

$$(e_1, e_2) \in R \text{ and } e_1 \stackrel{l}{\longrightarrow}_c e_1' \quad \text{imply} \quad e_2 \stackrel{l}{\longrightarrow}_c e_2' \text{ and } e_1' \sim_c \circ R \circ \sim_c e_2', \text{ for some } e_2'. \tag{9}$$

We shall now prove (9). Assume that $(e_1, e_2) \in R$. By the definition of R, this is because

(i) $e_1 = (\prod_{h \in H}(\{n_h\} \Rightarrow t_h))|(\varnothing \Rightarrow t)$ for some $\{n_h \mid h \in H\} \subseteq I\!N_+$, $\{t_h \mid h \in H\} \subseteq \mathbf{P}$, and $t \in \mathbf{P}$,

(ii) $e_2 = (\prod_{k \in K}(\{m_k\} \Rightarrow r_k))|(\varnothing \Rightarrow r)$ for some $\{m_k \mid k \in K\} \subseteq I\!N_+$, $\{r_k \mid k \in K\} \subseteq \mathbf{P}$, and $r \in \mathbf{P}$, and

(iii) there exists $x = \prod_{i \in I}(\{n_i\} \Rightarrow p_i)$ such that $e_1|x \sim_c e_2|x$ and

$$(\{n_h \mid h \in H\} \cup \{m_k \mid k \in K\}) \cap \{n_i \mid i \in I\} = \varnothing.$$

Suppose now that $e_1 \stackrel{l}{\longrightarrow}_c e_1'$. By (i), this move may take one of two possible forms:

- $e_1 \stackrel{l}{\longrightarrow}_c e_1'$ because, wlog, $(\{n_{h_1}\} \Rightarrow t_{h_1}) \stackrel{l}{\longrightarrow}_c f$ and $e_1' = f|(\prod_{h \in H - \{h_1\}}(\{n_h + 1\} \Rightarrow t_h))|(\varnothing \Rightarrow t)$, or
- $e_1 \stackrel{l}{\longrightarrow}_c e_1'$ because $(\varnothing \Rightarrow t) \stackrel{l}{\longrightarrow}_c f$ and $e_1' = (\prod_{h \in H}(\{n_h + 1\} \Rightarrow t_h))|f$.

We proceed by examining the two possibilities separately.

- Assume that $e_1 \stackrel{l}{\longrightarrow}_c e_1'$ because, wlog, $(\{n_{h_1}\} \Rightarrow t_{h_1}) \stackrel{l}{\longrightarrow}_c f$ and

  $$e_1' = f|(\prod_{h \in H - \{h_1\}}(\{n_h + 1\} \Rightarrow t_h))|(\varnothing \Rightarrow t).$$

  (Recall that δ has the effect of incrementing the indices in the sub-terms not involved in the derivation.) By Lemma 7.1(2), $l = (a, \{n_{h_1}\})$, for some $a \in A$, and $f \sim_c (\{1\} \Rightarrow y)|(\{n_{h_1} + 1\} \Rightarrow z)$, for some $y, z \in \mathbf{P}$. Hence, by substitutivity,

  $$e_1' \sim_c (\{1\} \Rightarrow y)|(\{n_{h_1} + 1\} \Rightarrow z)|(\prod_{h \in H - \{h_1\}}(\{n_h + 1\} \Rightarrow t_h))|(\varnothing \Rightarrow t) = g.$$

  By the operational semantics, $e_1 \stackrel{(a, \{n_{h_1}\})}{\longrightarrow}_c e_1'$ implies

  $$e_1|x \stackrel{(a, \{n_{h_1}\})}{\longrightarrow}_c e_1'|\delta(x) \sim_c g|\prod_{i \in I}(\{n_i + 1\} \Rightarrow p_i).$$

  As $e_1|x \sim_c e_2|x$, there exists $\hat{x}$ such that $e_2|x \stackrel{l}{\longrightarrow}_c \hat{x}$ and $e_1'|\delta(x) \sim_c \hat{x}$. Recall that $l = (a, n_{h_1})$ and that $n_{h_1} \notin \{n_i \mid i \in I\}$ by (iii). Hence, by Lemma 7.1, we may assume, wlog, that $e_2|x \stackrel{l}{\longrightarrow}_c \hat{x}$ because $e_2 \stackrel{l}{\longrightarrow}_c e_2'$ and $\hat{x} = e_2'|\delta(x)$, for some e_2'. Moreover, $e_2 \stackrel{l}{\longrightarrow}_c e_2'$ because, wlog, $m_{k_1} = n_{h_1}$, $(m_{k_1} \Rightarrow r_{k_1}) \stackrel{l}{\longrightarrow}_c f'$ and

  $$e_2' = f'|(\prod_{k \in K - \{k_1\}}(\{m_k + 1\} \Rightarrow r_k))|(\varnothing \Rightarrow r).$$

  By Lemma 7.1(2), $f' \sim_c (\{1\} \Rightarrow y')|(\{m_{k_1} + 1\} \Rightarrow z')$, for some $y', z' \in \mathbf{P}$. Hence,

  $$e_2' \sim_c (\{1\} \Rightarrow y')|(\{m_{k_1} + 1\} \Rightarrow z')|(\prod_{k \in K - \{k_1\}}(\{m_k + 1\} \Rightarrow r_k))|(\varnothing \Rightarrow r) = g'.$$

Now, as $g|\delta(x) \sim_c g'|\delta(x), \delta(x) = \prod_{i\in I}(\{n_i+1\} \Rightarrow p_i)$ and $(\{1\} \cup \{n_h+1 \mid h \in H\} \cup \{m_k+1 \mid k \in K\}) \cap \{n_i+1 \mid i \in I\} = \varnothing$, we have that

$$e_1' \sim_c g\, R\, g' \sim_c e_2'.$$

Hence $e_1' \sim_c \circ R \circ \sim_c e_2'$ and we have checked that (9) holds in this case.

- Assume that $e_1 \xrightarrow{l}_c e_1'$ because $(\varnothing \Rightarrow t) \xrightarrow{l}_c f$ and $e_1' = (\prod_{h\in H}(\{n_h+1\} \Rightarrow t_h))|f$. By Lemma 7.1(1), $l = (a, \varnothing)$ for some $a \in A$ and $f \sim_c (\{1\} \Rightarrow y)|(\varnothing \Rightarrow z)$ for some $y, z \in \mathbf{P}$. Hence, by substitutivity and Fact 5.1,

$$e_1' \sim_c (\{1\} \Rightarrow y)|(\prod_{h\in H}(\{n_h+1\} \Rightarrow t_h))|(\varnothing \Rightarrow z) = g.$$

By the operational semantics, $e_1 \xrightarrow{(a,\varnothing)}_c e_1'$ implies

$$e_1|x \xrightarrow{(a,\varnothing)}_c e_1'|\delta(x) \sim_c g|\prod_{i\in I}(\{n_i+1\} \Rightarrow p_i).$$

As $e_1|x \sim_c e_2|x$, there exists $\hat{x}$ such that $e_2|x \xrightarrow{(a,\varnothing)}_c \hat{x}$ and $e_1'|\delta(x) \sim_c \hat{x}$. By Lemma 7.1, it must be the case that $e_2|x \xrightarrow{(a,\varnothing)}_c \hat{x}$ because $e_2 \xrightarrow{(a,\varnothing)}_c e_2'$ and $\hat{x} = e_2'|\delta(x)$, for some e_2'. Moreover, again by Lemma 7.1, $e_2 \xrightarrow{(a,\varnothing)}_c e_2'$ because $(\varnothing \Rightarrow r) \xrightarrow{(a,\varnothing)}_c f'$ and $e_2' = (\prod_{k\in K}(\{m_k+1\} \Rightarrow r_k))|f'$. By Lemma 7.1(1), $f' \sim_c (\{1\} \Rightarrow y')|(\varnothing \Rightarrow z')$, for some $y', z' \in \mathbf{P}$. Hence we have that

$$e_2' \sim_c (\prod_{k\in K}(\{m_k+1\} \Rightarrow r_k))|(\{1\} \Rightarrow y')|(\varnothing \Rightarrow z') = g'.$$

Now, as $g|\delta(x) \sim_c g'|\delta(x)$ and $(\{1\} \cup \{n_h+1 \mid h \in H\} \cup \{m_k+1 \mid k \in K\}) \cap \{n_i+1 \mid i \in I\} = \varnothing$, we have that

$$e_1' \sim_c g\, R\, g' \sim_c e_2'.$$

Hence $e_1' \sim_c \circ R \circ \sim_c e_2'$ and we have checked that (9) holds in this case.

As we have examined all the possible origins of the move $e_1 \xrightarrow{l}_c e_1'$, the proof of (9) and hence of the lemma is complete. □

Appendix C: Proof of the Parallel Decomposition Theorem

Proof of Theorem 7.1: We prove the statement by induction on $\|(\prod_{i\in I}(\{n_i\} \Rightarrow p_i))|(\varnothing \Rightarrow r)\|$. (Recall that, by Fact 5.1, $x \sim_c y$ implies $\|x\| = \|y\|$.) We assume, as inductive hypothesis, that the claim is true for all $x, y \in \mathbf{P}_c$ of the required form such that $x \sim_c y$ and $\|x\| < \|(\prod_{i\in I}(\{n_i\} \Rightarrow p_i))|(\varnothing \Rightarrow r)\|$. We proceed to show that the claim holds for

$$e_1 = (\prod_{i\in I}(\{n_i\} \Rightarrow p_i))|(\varnothing \Rightarrow r) \quad \sim_c \quad (\prod_{j\in J}(\{m_j\} \Rightarrow q_j))|(\varnothing \Rightarrow t) = e_2. \tag{10}$$

The proof of this statement proceeds by a case analysis, depending on whether $\|(\varnothing \Rightarrow r)\| = 0$ or not.

- Assume that $\|(\varnothing \Rightarrow r)\| = 0$, i.e. $r \sim_c NIL$. The first thing to note is that we also have $t \sim_c NIL$, otherwise e_2 would be able to perform an action of the form $(a, \varnothing)$, for some

$a \in A$, whilst, by Lemma 7.1, e_1 can not. Hence $r \sim_c t$ and, by Fact 5.1, (10) then reduces to

$$f_1 = \prod_{i \in I}(\{n_i\} \Rightarrow p_i) \quad \sim_c \quad \prod_{j \in J}(\{m_j\} \Rightarrow q_j) = f_2. \tag{11}$$

Obviously, if $I = \varnothing$ then $J = \varnothing$. In this case, $\phi = \varnothing$ is the required bijection showing that $\{(\{n_i\} \Rightarrow p_i) \mid i \in \varnothing\} \sim_c \{(\{m_j\} \Rightarrow q_j) \mid j \in \varnothing\}$. This is the basis of the inductive argument.

Assume then that $I \neq \varnothing$ and $(\{n_{i_1}\} \Rightarrow p_{i_1}) \stackrel{l}{\longrightarrow}_c x$ for some x. By Lemma 7.1(2), l has the form $(a, \{n_{i_1}\})$, for some $a \in A$, and $x \sim_c (\{1\} \Rightarrow y)|(\{n_{i_1}+1\} \Rightarrow z)$, for some $y, z \in \mathbf{P}$. By the operational semantics, $(\{n_{i_1}\} \Rightarrow p_{i_1}) \stackrel{l}{\longrightarrow}_c x$ implies

$$\prod_{i \in I}(\{n_i\} \Rightarrow p_i) \stackrel{l}{\longrightarrow} x|(\prod_{i \in I-\{i_1\}}(\{n_i+1\} \Rightarrow p_i)) = f_1'.$$

Moreover, by substitutivity,

$$x|(\prod_{i \in I-\{i_1\}}(\{n_i+1\} \Rightarrow p_i)) \sim_c (\{1\} \Rightarrow y)|(\{n_{i_1}+1\} \Rightarrow z)|(\prod_{i \in I-\{i_1\}}(\{n_i+1\} \Rightarrow p_i)).$$

As $f_1 \sim_c f_2$, there exists $f_2' \in \mathbf{P}_c$ such that $f_2 \stackrel{l}{\longrightarrow}_c f_2'$ and $f_2' \sim_c f_1'$. Suppose, wlog, that $f_2 \stackrel{l}{\longrightarrow}_c f_2'$ because $(\{m_{j_1}\} \Rightarrow q_{j_1}) \stackrel{l}{\longrightarrow}_c x'$ and $f_2' = x'|(\prod_{j \in J-\{j_1\}}(\{m_j+1\} \Rightarrow q_j))$. By Lemma 7.1, $n_{i_1} = m_{j_1}$ and $x' \sim_c (\{1\} \Rightarrow y')|(\{m_{j_1}+1\} \Rightarrow z')$, for some $y', z' \in \mathbf{P}$. Thus we have that

$$f_1' \quad \sim_c \quad (\{1\} \Rightarrow y)|(\{n_{i_1}+1\} \Rightarrow z)|(\prod_{i \in I-\{i_1\}}(\{n_i+1\} \Rightarrow p_i)) \tag{12}$$

$$\sim_c \quad (\{1\} \Rightarrow y')|(\{m_{j_1}+1\} \Rightarrow z')|(\prod_{j \in J-\{j_1\}}(\{m_j+1\} \Rightarrow q_j)) \tag{13}$$

$$\sim_c \quad f_2'. \tag{14}$$

Note that $\| f_1' \| < \| e_1 \|$. Moreover, the index sets $\{n_i+1 \mid i \in I\} \cup \{1\}$ and $\{m_j+1 \mid j \in J\} \cup \{1\}$ satisfy the proviso of the theorem. It is also easy to see that

- $y \sim_c NIL$ iff $y' \sim_c NIL$ and
- $z \sim_c NIL$ iff $z' \sim_c NIL$. (Otherwise, by Lemma 7.1(2) and the assumption on the index sets, the two terms f_1' and f_2' would not even be trace equivalent.)

Hence, modulo possible applications of Fact 5.1(3), we may apply induction to (12)-(13) to obtain at least that

$$\{(\{n_i+1\} \Rightarrow p_i) \mid i \in I - \{i_1\}\} \sim_c \{(\{m_j+1\} \Rightarrow q_j) \mid j \in J - \{j_1\}\}.$$

Obviously, we also have that

$$\{(\{n_i\} \Rightarrow p_i) \mid i \in I - \{i_1\}\} \sim_c \{(\{m_j\} \Rightarrow q_j) \mid j \in J - \{j_1\}\}.$$

By substitutivity, this implies that $\prod_{i \in I-\{i_1\}}(\{n_i\} \Rightarrow p_i) \sim_c \prod_{j \in J-\{j_1\}}(\{m_j\} \Rightarrow q_j)$ and, by substituting in (11), we get that

$$(\{n_{i_1}\} \Rightarrow p_{i_1})|(\prod_{i \in I-\{i_1\}}(\{n_i\} \Rightarrow p_i)) \quad \sim_c \quad (\{m_{j_1}\} \Rightarrow q_{j_1})|(\prod_{i \in I-\{i_1\}}(\{n_i\} \Rightarrow p_i)). \tag{15}$$

Recall now that $n_{i_1} = m_{j_1}$ and $n_{i_1} \notin \{n_i \mid i \in I - \{i_1\}\}$ by the hypotheses of the theorem. We may then apply the first statement of the Simplification Lemma to (15) to obtain that $(\{n_{i_1}\} \Rightarrow p_{i_1}) \sim_c (\{m_{j_1}\} \Rightarrow q_{j_1})$. We already know that $n_{i_1} = m_{j_1}$. Moreover, by Proposition 7.1(3), we also have that $p_{i_1} \sim_c q_{j_1}$. Hence

$$\{(\{n_i\} \Rightarrow p_i) \mid i \in I\} \sim_c \{(\{m_j\} \Rightarrow q_j) \mid j \in J\}.$$

This completes the proof of the claim for this sub-case.

- Assume that $\|(\varnothing \Rightarrow r)\| > 0$, i.e. $r \not\sim_c NIL$. Reasoning as in the previous case we also have that $t \not\sim_c NIL$. As $\|(\varnothing \Rightarrow r)\| > 0$, there exist $l \in L$ and $x \in \mathbf{P}_c$ such that $(\varnothing \Rightarrow r) \stackrel{l}{\longrightarrow}_c x$. By Lemma 7.1(1), $l = (a, \varnothing)$, for some $a \in A$, and $x \sim_c (\{1\} \Rightarrow y)|(\varnothing \Rightarrow z)$, for some $y, z \in \mathbf{P}$. By the operational semantics, $(\varnothing \Rightarrow r) \stackrel{l}{\longrightarrow}_c x$ implies

$$e_1 \stackrel{l}{\longrightarrow}_c (\prod_{i \in I}(\{n_i + 1\} \Rightarrow p_i))|x = e_1'.$$

By substitutivity, we have that $e_1' \sim_c (\prod_{i \in I}(\{n_i + 1\} \Rightarrow p_i))|(\{1\} \Rightarrow y)|(\varnothing \Rightarrow z)$.

As $e_1 \sim_c e_2$, there exists e_2' such that $e_2 \stackrel{l}{\longrightarrow}_c e_2'$ and $e_1' \sim_c e_2'$. By Lemma 7.1, as $l = (a, \varnothing)$, it must be the case that $e_2 \stackrel{l}{\longrightarrow}_c e_2'$ because, for some x', $(\varnothing \Rightarrow t) \stackrel{l}{\longrightarrow}_c x'$ and $e_2' = (\prod_{j \in J}(\{m_j + 1\} \Rightarrow q_j))|x'$. By Lemma 7.1(1), $x' \sim_c (\{1\} \Rightarrow y')|(\varnothing \Rightarrow z')$, for some $y', z' \in \mathbf{P}$. Thus we have that

$$\begin{aligned} e_1' &\sim_c (\prod_{i \in I}(\{n_i + 1\} \Rightarrow p_i))|(\{1\} \Rightarrow y)|(\varnothing \Rightarrow z) && (16)\\ &\sim_c (\prod_{j \in J}(\{m_j + 1\} \Rightarrow q_j))|(\{1\} \Rightarrow y')|(\varnothing \Rightarrow z') && (17)\\ &\sim_c e_2'. && (18) \end{aligned}$$

Note that $\|e_1'\| < \|e_1\|$. Moreover the index sets $\{n_i + 1 \mid i \in I\} \cup \{1\}$ and $\{m_j + 1 \mid j \in J\} \cup \{1\}$ satisfy the proviso of the theorem. Again using Lemma 7.1, it is easy to see that $y \sim_c NIL$ iff $y' \sim_c NIL$. Hence, modulo a possible application of Fact 5.1(3), we may apply induction to (16)-(17) to obtain at least that

$$\{(\{n_i + 1\} \Rightarrow p_i) \mid i \in I\} \sim_c \{(\{m_j + 1\} \Rightarrow q_j) \mid j \in J\}.$$

Obviously, we also have that

$$\{(\{n_i\} \Rightarrow p_i) \mid i \in I\} \sim_c \{(\{m_j\} \Rightarrow q_j) \mid j \in J\}.$$

By substitutivity, this implies that

$$\prod_{i \in I}(\{n_i\} \Rightarrow p_i) \sim_c \prod_{j \in J}(\{m_j\} \Rightarrow q_j).$$

By substituting in (10) we have that

$$(\prod_{i \in I}(\{n_i\} \Rightarrow p_i))|(\varnothing \Rightarrow r) \sim_c (\prod_{i \in I}(\{n_i\} \Rightarrow p_i))|(\varnothing \Rightarrow t). \quad (19)$$

We may now apply the second statement of the Substitution Lemma to (19) to obtain $(\varnothing \Rightarrow r) \sim_c (\varnothing \Rightarrow t)$, i.e. $r \sim_c t$.

This completes the inductive argument and the proof of the theorem. □