Observing distribution in processes: static and dynamic localities

Ilaria Castellani

N° 2276

Mai 1994

PROGRAMME 2

Calcul symbolique,

programmation

et génie logiciel

Observing distribution in processes: static and dynamic localities

Ilaria Castellani

Programme 2 — Calcul symbolique, programmation et génie logiciel
Projet MEIJE

Rapport de recherche n° 2276 — Mai 1994 — 45 pages

Abstract: The distributed structure of CCS processes can be made explicit by assigning different *locations* to their parallel components. The assignment of locations may be done statically, or dynamically as the execution proceeds. The dynamic approach was developed first, by Boudol *et al.*, as it appeared more convenient for defining notions of *location equivalence* and *preorder*. Extending previous work by L. Aceto we study here the static approach, which is more natural from an intuitive point of view, and more manageable for verification purposes. We define static notions of location equivalence and preorder, and show that they coincide with the dynamic ones. To establish the equivalence of the two location semantics, we introduce an intermediate transition system called *occurrence system*, which incorporates both notions of locality. This system supports a definition of *local history preserving bisimulation* for CCS, which is a third formulation of location equivalence.

Key-words: Concurrent and distributed processes, CCS, bisimulation equivalence, noninterleaving semantics, locations, location equivalence and preorder, causal semantics, history-preserving bisimulation.

(Résumé : tsvp)

This work has been partly supported by the Project 502-1 of the Indo-French Centre for the Promotion of Advanced Research.

Unité de recherche INRIA Sophia-Antipolis
2004 route des Lucioles, BP 93, 06902 SOPHIA-ANTIPOLIS Cedex (France)
Téléphone : (33) 93 65 77 77 – Télécopie : (33) 93 65 77 65

Observation de la répartition des processus: localités statiques et dynamiques

Résumé : On peut rendre explicite la structure de répartition d'un processus CCS en attribuant des localités différentes à ses composants parallèles. L'attribution des localités peut se faire de façon statique, ou dynamiquement au cours de l'execution. L'approche dynamique a été développée en premier, par Boudol *et al.*, se prêtant mieux à la définition de notions d'*équivalence* et de *préordre de répartition*. Etendant un travail de L. Aceto nous étudions ici l'approche statique, qui est plus proche de l'intuition et mieux adaptée à des fins de vérification. Nous définissons des notions statiques d'équivalence et de préordre de répartition, et nous montrons qu'elles coïncident avec les notions dynamiques. Pour établir l'égalité des deux sémantiques, nous introduisons un système de transitions intermédiaire appelé *système d'occurrences*, qui incorpore les deux notions de localité. Ce système se prête aussi à la définition d'une *bisimulation préservant l'histoire locale*, qui est une troisième formulation de l'équivalence de répartition.

Mots-clé : Processus parallèles et répartis, CCS, bisimulation, parallélisme vrai, localités, équivalence et préordre de répartition, sémantique causale, bisimulation préservant l'histoire.

1 Introduction

This work is concerned with *distributed semantics* for CCS, accounting for the spatial distribution of processes. Such semantics focus on different aspects of behaviour than most non-interleaving semantics for CCS considered so far in the literature, which are based on the notion of causality. Roughly speaking, a distributed semantics keeps track of the behaviour of the local components of a system, and thus is appropriate for describing phenomena like a local deadlock. On the other hand a causal semantics, such as those described in [DDNM89], [GG89], [DD90] [BC91], is concerned with the flow of causality among activities and thus is better suited to model the interaction of processes and the global control structure of a system.

The distributed structure of CCS processes can be made explicit by assigning different *locations* to their parallel components. To this end we use a *location prefixing* construct $l :: p$ [BCHK93], [BCHK91], which represents process p residing at location l. The actions of such a process are observed together with their location. We have for instance:

$$(l :: a \mid k :: b) \xrightarrow[l]{a} (l :: nil \mid k :: b) \xrightarrow[k]{b} (l :: nil \mid k :: nil)$$

In general, because of the nesting of parallelism, the locations of actions will not be simple letters $l, k, \ldots$ but rather words $u = l_1 \cdots l_n$. Then a "distributed process" will perform transitions of the form $p \xrightarrow[u]{a} p'$.

Intuitively, the assignment of locations should be done statically, and then become part of what is observed of a process. More precisely, CCS processes should be observed through their *distributions*, which are obtained by transforming each subprocess $(p \mid q)$ into $(l :: p \mid k :: q)$, where l and k are distinct locations. When comparing the behaviours of processes, this will allow us for instance to distinguish $(a \mid b)$ from $(a.\, b + b.\, a)$, since any distribution of the first process will perform actions a and b at different locations. For more interesting examples we refer the reader to the introductions of [BCHK93], [BCHK91].

The question is now: which notion of *abstract distributed behaviour* do these transitions induce on CCS processes? More specifically, we look for a notion of weak bisimulation based on the transitions $p \xrightarrow[u]{a} p'$. Roughly speaking, this should equate processes exhibiting the same "location transitions". In our view, however, it would be too strong a requirement to ask for the identity of locations in corresponding transitions. In fact, if we want to observe distribution, we still aim, to some extent, at an extensional semantics. For instance, we do not want to observe the order in which parallel components have been assembled in a system, nor indeed the number of these components. We are only interested in the number of *active* components in each computation. We would like e.g. to identify (the distributions of) the CCS processes:

$$\begin{array}{ccc} a \mid (b \mid c) & \text{and} & (a \mid b) \mid c \\ a & \text{and} & a \mid nil \end{array}$$

To this end, transitions must be compared modulo an *association* between their locations. For instance to relate the distributions $l :: a \mid k :: (l' :: b \mid k' :: c)$ and $l :: (l' :: a \mid k' :: b) \mid k :: c$, we need to identify the locations $l,\ kl',\ kk'$ of the first respectively with $ll',\ lk',\ k$ in the second. However, this association cannot in general be fixed statically. For consider the two processes:

$$p = [(\alpha + b) \mid \bar{\alpha}.\, b] \backslash \alpha \quad \text{and} \quad q = b$$

Intuitively, we would like to equate p and q because the observable behaviour of any distributions of these processes consists in just one action b at some location l. But here the required association of locations will depend on which computation is chosen in the first process. Hence it is not immediately clear how to define an equivalence based on static locations.

Because of this difficulty, the static approach was initially abandoned in favour of a different one, where locations are introduced dynamically as the execution proceeds. This *dynamic approach*, where locations are associated with actions rather than with parallel components, has been presented in [BCHK93], [BCHK91]. In this setting, the choice of locations is more flexible and the notion of *location equivalence* is particularly simple: it is just the standard notion of bisimulation, applied to the transitions $p \xrightarrow[u]{a} p'$. Moreover, by weakening a little the definition of the equivalence, we obtain a notion of *location preorder*, which formalises the idea that one process is more sequential or less distributed than another. Such a notion is particularly useful when dealing with truly concurrent semantics, where an implementation is often not equivalent to its specification. Since location equivalence and preorder are essentially bisimulation relations, many proof techniques familiar from the theory of standard bisimulation may be applied to them: for example both these relations have a complete axiomatisation and a logical characterisation in the style of Hennessy and Milner, see [BCHK93], [BCHK91].

However, the dynamic approach has the drawback of yielding infinite transition systems even for regular processes, and thus cannot be directly used for verification purposes. Moreover in this approach locations represent access paths for actions rather than sites in a system, and thus are somehow remote from the original intuition. For these reasons, it was interesting to resume the initial attempt at a static approach. The problem of finding the appropriate notion of bisimulation was solved by L. Aceto in [Ace91] for *nets of automata*, a subset of CCS where parallelism is only allowed at the top level. The key idea here is to replace the usual notion of a bisimulation relation by that of a family of relations indexed by increasing location associations (what we call here a *progressive bisimulation family*). Aceto shows that the notions of static location equivalence and preorder thus obtained coincide with the dynamic ones, and thus may be used as “effective” versions of the latter.

The purpose of the present work is to generalize the static treatment of Aceto to full CCS. Having established the notion of distribution for general CCS processes, the main point is to adapt Aceto's definitions of static location equivalence and preorder. Because of the arbitrary nesting of parallelism and prefixing in CCS terms, and of the interplay between sum and parallelism, this is not completely straightforward. A step in this direction was done by Mukund and Nielsen in [MN92], where a notion of bisimulation equivalence based on static locations is proposed for a class of asynchronous transition systems modelling CCS with guarded sums. The notion of equivalence we present here is essentially the same (extended to all CCS), and our main result is that it coincides with the dynamic location equivalence of [BCHK91]. We also give a similar result for the location preorders.

To compare the two location semantics we introduce an intermediate transition system, called *occurrence transition system*, which incorporates both the static and dynamic locations. This system has an interest of its own, as it allows for a precise definition of the notion of *occurrence* of an action in a computation. It also supports a syntactic definition of *(local) history preserving bisimulation*, which turns out to be another formulation of location equivalence.

We conclude this introduction with a brief review of related work. A first distributed semantics for a subset of CCS was proposed in [CH89], [Cas88], where the notion of *distributed bisimulation* was introduced. An extension of this notion to a larger subset of CCS was investigated by A. Kiehn in [Kie89]. A variant of Kiehn's extension is examined in [CN93]. Concerning distributed bisimulation, we should mention also the decidability result of [Chr92], for a recursive fragment of CCS with parallelism. The precise relation between distributed bisimulation and (dynamic) location equivalence is studied in [BCHK93] and [BCHK91]; let us just mention here that Kiehn's extended notion is weaker than location bisimulation.

A general comparison of distributed and causality-based semantics is carried out in [Kie91]; in particular Kiehn gives a characterization of dynamic location equivalence as a *local cause bisimulation*, a variant of the causal bisimulation of [DD90] based on local rather than global causality. A similar result is presented by Montanari and Yankelevich in [MY92], [Yan93], where dynamic location equivalence is characterised as a *local mixed-ordering equivalence*, a variant of the mixed-ordering equivalence of [DDNM89]. Our characterisation of location equivalence as a local history preserving bisimulation is therefore not surprising, since causal bisimulation, history preserving bisimulation and mixed-ordering equivalence are known (from [DD91], [Ace92]) to be different formulations of the same equivalence.

As regards the static approach to locations, we mentioned already that our work comes in the line of [Ace91] and [MN92]. A transition system for CCS labelled with static locations, called "spatial transition system", was considered also in [MY92], [Yan93]. However locations are used there essentially to build a second transition system, labelled by partial orders, which is then used for defining the local mixed-ordering equivalence. Again, this partial order transition system gives finite representations only for finite behaviours.

Finally, D. Murphy in a recent paper [Mur93] proposes a more concrete view of localities, for nets of finite sequential processes (essentially a sublanguage of that considered in [Ace91]). Here again localities are given statically. However the names of localities are themselves significant, and processes are considered equivalent only if they reside on the same set of localities and present the same behaviour at each locality. So Murphy's concern appears to be different from ours: he compares distributions of processes on a given set of localities (or processors), while our semantics reflects the notions of distributed behaviour and degree of distribution somewhat abstractly.

The paper is organised as follows. In Section 2 we introduce our language for processes with locations. Sections 3 and 4 present respectively the static and the dynamic location semantics. Section 5 is devoted to the comparison of the two approaches. Finally, in the Appendix, we give the definition of local history preserving bisimulation and present a finitely-branching transition system for the dynamic location semantics, inspired from [Yan93].

This is an extended version of [Cas93], complete with proofs.

2 A language for processes with locations

We introduce now a language for specifying processes with locations, called LCCS. This language is a simple extension of CCS, including a new construct to deal with locations.

Let us recall some conventions of CCS [Mil80]. One assumes a set of names Λ, ranged over by $\alpha, \beta, \ldots$, and a corresponding set of co-names $\bar{\Lambda} = \{\bar{\alpha} \mid \alpha \in \Lambda\}$, where $\bar{}$ is a bijection such that $\bar{\bar{\alpha}} = \alpha$ for all $\alpha \in \Lambda$. The set of visible actions is given by $Act = \Lambda \cup \bar{\Lambda}$. Invisible actions – representing internal communications – are denoted by the symbol $\tau \notin Act$. The set of all actions is then $Act_\tau =_{\text{def}} Act \cup \{\tau\}$. We use $a, b, c, \ldots$ to range over Act and $\mu, \nu, \ldots$ to range over Act_τ. We also assume a set V of process variables, ranged over by $x, y \ldots$.

In addition to the operators of CCS, which we suppose the reader to be familiar with, LCCS includes a construct for building processes with explicit locations. Let *Loc*, ranged over by $l, k, \ldots$, be an infinite set of atomic locations. The new construct of *location prefixing*, noted $l :: p$, is used to represent process p residing at location l. Intuitively, the actions of such a process will be observed "within location l". The syntax of LCCS is as follows:

$$p ::= nil \;\mid\; \mu.p \;\mid\; (p \mid q) \;\mid\; (p + q) \;\mid\; p\backslash\alpha \;\mid\; p\langle f\rangle \;\mid\; x \;\mid\; rec\,x.\,p \;\mid\; l :: p$$

Here we use the slightly nonstandard notation $p\langle f\rangle$ to represent the relabelling operator of CCS.

In a previous paper [BCHK91], this language has been given a location semantics based on a dynamic assignment of locations to processes. Here we shall present a location semantics based on a static notion of location, and show that the two approaches, dynamic and static, give rise to the same notions of equivalence and preorder on CCS processes. The basic idea, common to both approaches, is that the actions of processes are observed together with the locations at which they occur. In general, because of the nesting of parallelism and prefixing in terms, the locations of actions will not be atomic locations of *Loc*, but rather *words* over these locations. Thus general locations will be elements $u, v \dots$ of Loc^*, and processes will be interpreted as performing transitions $p \xrightarrow[u]{\mu} p'$, where μ is an action and u is the location at which it occurs.

However, locations do not have the same intuitive meaning in the two approaches. In the static approach locations represent sites - or parallel components - in a distributed system, much as one would expect. In the dynamic approach, on the other hand, the location of an action represents the sequence of actions which are locally necessary to enable it, and thus is more properly viewed as an access path to that action within the component where it occurs. Because of this difference in intuition, it is not immediately obvious that the two approaches should yield the same semantic notions. The fact that they do means that observing distribution is essentially the same as observing local causality.

3 Static approach

We start by presenting an operational semantics for LCCS based on the static notion of location. The idea of this semantics is very simple. Processes of LCCS have some components of the form $l :: p$, and the actions arising from these components are observed together with their location. The distribution of locations in a term remains fixed through execution. Location prefixing is a static construct and the operational rules do not create new locations; they simply exhibit the locations which are already present in terms. Formally, this is expressed by the operational rules for action prefixing and location prefixing. Recall that locations are words $u, v, \ldots \in Loc^*$. The empty word ε represents the location of the overall system. The rules for $\mu.p$ and $l :: p$ are respectively:

$$\text{(S1)} \qquad \mu.p \xrightarrow[\varepsilon]{\mu}_s p$$

$$\text{(S2)} \qquad p \xrightarrow[u]{\mu}_s p' \qquad \Rightarrow \qquad l :: p \xrightarrow[l\,u]{\mu}_s l :: p'$$

Rule (S1) says that an action which is not in the scope of any location l is observed as a global action of the system. Rule (S2) shows how locations are transferred from processes to actions. The rules for the remaining operators, apart from the communication rule, are similar to the standard interleaving rules for CCS, with transitions $\xrightarrow[u]{\mu}_s$ replacing the usual transitions $\xrightarrow{\mu}$.

The set of all rules specifying the operational semantics of LCCS is given in Figure 1. The rule for communication (S4) requires some explanation. In the strong location transition system we take the location of a communication to be that of the smallest component which includes the two communicating subprocesses: the notation $u \sqcap v$ in rule (S4) stands for the longest common prefix of u and v. For instance:

Example 3.1 *Let* $p = l :: \alpha \mid k :: \bar{\alpha}.\,(l' :: \beta \mid k' :: \bar{\beta})$, *with* $l \neq k,\ l' \neq k'$. *Then:*

$$p \ \xrightarrow[\varepsilon]{\tau}_s \ l :: nil \mid k :: (l' :: \beta \mid k' :: \bar{\beta}) \ \xrightarrow[k]{\tau}_s \ l :: nil \mid k :: (l' :: nil \mid k' :: nil)$$

However, we shall mostly be interested here in the *weak location transition system*, where τ-transitions will have no explicit location: since the transitions themselves are not observable, it would not make much sense to attribute a location to them. The weak location transitions $\overset{a}{\underset{u}{\Longrightarrow}}_s$ and $\overset{\tau}{\Longrightarrow}_s$ are thus defined by:

$$\begin{array}{lll} p \overset{\tau}{\Longrightarrow}_s q & \Leftrightarrow & \exists\, u_1, \ldots, u_n, p_0, \ldots, p_n \ \ s.t. \ \ p = p_0 \xrightarrow[u_1]{\tau}_s p_1 \cdots \xrightarrow[u_n]{\tau}_s p_n = q \\ p \overset{a}{\underset{u}{\Longrightarrow}}_s q & \Leftrightarrow & \exists\, p_1, p_2 \ \ s.t. \ \ p \overset{\tau}{\Longrightarrow}_s p_1 \xrightarrow[u]{a}_s p_2 \overset{\tau}{\Longrightarrow}_s q \end{array}$$

We shall use the weak location transition system as the basis for defining a new semantic theory for CCS, and in particular notions of equivalence and preorder which account for the degree of distribution of processes.

The reader may have noticed, however, that applying the rules of Figure 1 to CCS terms just yields a transition $p \xrightarrow[\varepsilon]{\mu}_s p'$ whenever the standard semantics yields a transition $p \xrightarrow{\mu} p'$. In fact, we shall not apply these rules directly to CCS terms. Instead, the idea is to first bring out the parallel structure of CCS terms by assigning locations to their parallel components, thus transforming them into particular LCCS terms which we call "distributed processes", and then execute these according to the given operational rules. The set DIS $\subseteq$ LCCS of *distributed processes* is given by the grammar:

$$p ::= nil \ \mid \ \mu.p \ \mid \ \underbrace{(l :: p \mid k :: q)}_{l \neq k} \ \mid \ (p + q) \ \mid \ p\backslash\alpha \ \mid \ p\langle f\rangle \ \mid \ x \ \mid \ rec\,x.\,p$$

Essentially, a distributed process is obtained by inserting a pair of distinct locations in a CCS term wherever there occurs a parallel operator. This is formalised by the notion of *distribution*, which we define next.

For each $\mu \in Act_\tau$, $u \in Loc^*$, let $\xrightarrow[u]{\mu}_s$ be the least relation $\xrightarrow[u]{\mu}$ on LCCS processes satisfying the following axiom and rules.

$$
\begin{array}{llcl}
\text{(S1)} & \mu.p \xrightarrow[\varepsilon]{\mu} p & & \\
\text{(S2)} & p \xrightarrow[u]{\mu} p' & \Rightarrow & l :: p \xrightarrow[lu]{\mu} l :: p' \\
\text{(S3)} & p \xrightarrow[u]{\mu} p' & \Rightarrow & p \mid q \xrightarrow[u]{\mu} p' \mid q \\
 & & & q \mid p \xrightarrow[u]{\mu} q \mid p' \\
\text{(S4)} & p \xrightarrow[u]{\alpha} p', \quad q \xrightarrow[v]{\bar{\alpha}} q' & \Rightarrow & p \mid q \xrightarrow[u \sqcap v]{\tau} p' \mid q' \\
\text{(S5)} & p \xrightarrow[u]{\mu} p' & \Rightarrow & p + q \xrightarrow[u]{\mu} p' \\
 & & & q + p \xrightarrow[u]{\mu} p' \\
\text{(S6)} & p \xrightarrow[u]{\mu} p', \quad \mu \notin \{\alpha, \bar{\alpha}\} & \Rightarrow & p \backslash \alpha \xrightarrow[u]{\mu} p' \backslash \alpha \\
\text{(S7)} & p \xrightarrow[u]{\mu} p' & \Rightarrow & p \langle f \rangle \xrightarrow[u]{f(\mu)} p' \langle f \rangle \\
\text{(S8)} & p[rec\, x.\ p/x] \xrightarrow[u]{\mu} p' & \Rightarrow & rec\, x.\ p \xrightarrow[u]{\mu} p'
\end{array}
$$

Figure 1: Static location transitions

Let $p \xrightarrow[u]{\tau}_d q \quad \Leftrightarrow_{\text{def}} \quad p \xrightarrow[u]{\tau}_s q$, and for each $a \in Act$, $u \in Loc^*$, let $\xrightarrow[u]{a}_d$ be the least relation $\xrightarrow[u]{a}$ on LCCS processes satisfying rules (S2), (S3), (S5), (S6), (S7), (S8) and the axiom:

$$
\text{(D1)} \qquad a.p \xrightarrow[l]{a} l :: p \qquad \text{for any } l \in Loc
$$

Figure 2: Dynamic location transitions

Definition 3.2 The *distribution relation* is the least relation $\mathcal{D} \subseteq$ (CCS $\times$ DIS) satisfying:

- $nil\,\mathcal{D}\,nil$ and $x\,\mathcal{D}\,x$

- $p\,\mathcal{D}\,r \;\Rightarrow\;$ $\mu.p\;\mathcal{D}\;\mu.r$
  $p\backslash\alpha\;\mathcal{D}\;r\backslash\alpha$
  $p\,\langle f\rangle\;\mathcal{D}\;r\,\langle f\rangle$
  $(rec\,x.\,p)\;\mathcal{D}\;(rec\,x.\,r)$

- $p\,\mathcal{D}\,r\;\&\;q\,\mathcal{D}\,s \;\Rightarrow\;$ $(p \mid q)\;\mathcal{D}\;(l::r \mid k::s)\,,\;\; \forall\, l,k$ s.t. $l \neq k$
  $(p+q)\;\mathcal{D}\;(r+s)$

If $p\,\mathcal{D}\,r$ we say that r is a *distribution* of p.

Note that the same pair of locations may be used more than once in a distribution. We shall see in fact, at the end of this section, that distributions involving just *two* atomic locations are sufficient for describing the distributed behaviour of CCS processes.

3.1 Static location equivalence

We want to define an equivalence relation $\approx_\ell^s$ on CCS processes, based on a bisimulation-like relation between their distributions. The intuition for two CCS processes p, q to be equivalent is that there exist two distributions of them, say $\bar{p}$ and $\bar{q}$, which perform "the same" location transitions at each step. However, as we argued already in the introduction, we cannot require the identity of locations in corresponding transitions. If we want to identify the following processes:

$$a \mid (b \mid c) \qquad \text{and} \qquad (a \mid b) \mid c$$
$$a \qquad \text{and} \qquad a \mid nil$$

it is clear that, whatever distributions we choose, we must allow corresponding transitions to have different – although somehow related – static locations. In general transitions will be compared modulo an *association* between their locations. The idea is directly inspired from that used by Aceto for nets of automata [Ace91]; however in our case the association will not be a bijection as in [Ace91], nor even a function. For example, in order to equate the two processes:

$$a.\,(b.\,c \mid nil) \qquad \text{and} \qquad a.\,b.\,(c \mid nil)$$

we need an association containing the three pairs $(\varepsilon,\varepsilon), (l,\varepsilon), (l,l')$, for some $l, l' \in Loc$.

In fact, the only property we will require of location associations is that they respect independence of locations. To make this precise, let $\ll$ denote the prefix ordering on Loc^*. If $u \ll v$ we say that v is an extension or a *sublocation* of u. If $u \not\ll v$ and $v \not\ll u$, what we indicate by $u \diamond v$, we say that u and v are *independent*.

Definition 3.3 A relation $\varphi \subseteq (Loc^* \times Loc^*)$ is a *consistent location association (cla)* if:

$$(u,v) \in \varphi \quad \& \quad (u',v') \in \varphi \quad \Rightarrow \quad (u \diamond u' \;\Leftrightarrow\; v \diamond v')$$

Essentially the same notion of consistent association has been proposed by Mukund and Nielsen in [MN92] for a class of asynchronous transition systems modelling CCS with guarded sums. The following properties of *cla's* are straightforward to check:

Property 3.4 (Properties of cla's)

1. *If φ is a cla, then φ^{-1} is a cla.*
2. *If φ is a cla and $\psi \subseteq \varphi$, then ψ is a cla.*
3. *If φ and ψ are cla's, then $\varphi \circ \psi$ is a cla.*

Now Aceto showed in [Ace91] that, for a given pair of distributed processes we want to equate, the required *cla* cannot in general be fixed statically, but has to be built incrementally. For consider the two distributed processes, which are intuitively equivalent since both perform actions a and b in either order at different locations:

$$(\, l :: (a.\gamma + b.\bar{\gamma}) \mid k :: (\bar{\gamma}.b + \gamma.a)\,) \backslash \gamma \qquad \text{and} \qquad (l :: a \mid k :: b)$$

Here, depending on which summand is chosen in the left component of the first process, one will use the association $\varphi = \{(l,l),(k,k)\}$ or the association $\varphi' = \{(l,k),(k,l)\}$ (note that $\varphi \cup \varphi'$ is not consistent). Another example is given in the introduction.

To dynamically build up associations, we use the same technique as in [Ace91]. Let Φ be the set of consistent location associations. We define particular Φ-indexed families of relations S_φ over distributed processes, which we call *progressive bisimulation families* (although the relations that constitute a family are not themselves bisimulations). The idea is to start with the empty association of locations and extend it consistently as the bisimulation proceeds.

Definition 3.5 A *progressive bisimulation family (pbf)* is a Φ-indexed family $\mathbf{S} = \{S_\varphi \mid \varphi \in \Phi\}$ of relations over DIS such that, if $pS_\varphi q$ then for all $a \in Act, u \in Loc^*$:

(1) $p \overset{a}{\underset{u}{\Longrightarrow}}_s p' \;\Rightarrow\; \exists\, q', v$ s.t. $q \overset{a}{\underset{v}{\Longrightarrow}}_s q'$, $\varphi \cup \{(u,v)\} \in \Phi$ and $p' \, S_{\varphi\cup\{(u,v)\}}\, q'$

(2) $q \overset{a}{\underset{v}{\Longrightarrow}}_s q' \;\Rightarrow\; \exists\, p', u$ s.t. $p \overset{a}{\underset{u}{\Longrightarrow}}_s p'$, $\varphi \cup \{(u,v)\} \in \Phi$ and $p' \, S_{\varphi\cup\{(u,v)\}}\, q'$

(3) $p \overset{\tau}{\Longrightarrow}_s p' \;\Rightarrow\; \exists\, q'$ s.t. $q \overset{\tau}{\Longrightarrow}_s q'$ and $p' S_\varphi q'$

(4) $q \overset{\tau}{\Longrightarrow}_s q' \;\Rightarrow\; \exists\, p'$ s.t. $p \overset{\tau}{\Longrightarrow}_s p'$ and $p' S_\varphi q'$

We may now define the *location equivalence* $\approx_\ell^s$ on CCS terms as follows:

Definition 3.6 (**Static location equivalence**) For $p, q \in$ CCS, we let $p \approx_\ell^s q$ if and only if for some $\bar{p}, \bar{q} \in$ DIS such that $p\,\mathcal{D}\,\bar{p}$ and $q\,\mathcal{D}\,\bar{q}$, there exists a progressive bisimulation family $\mathbf{S} = \{S_\varphi \mid \varphi \in \Phi\}$ such that $\bar{p}\, S_\emptyset\, \bar{q}$.

We prove now that $\approx_\ell^s q$ is indeed an equivalence relation. The reader may have noticed that the inverse $\mathcal{D}^{-1}$ of the distribution relation is a function. If we let $\pi =_{\rm def} \mathcal{D}^{-1}$, then $\pi(p)$ gives the CCS process underlying the distributed process p. We start by showing that all distributions of the same process are in the relation $S_\emptyset$ for some progressive bisimulation family $\mathbf{S}$:

Proposition 3.7 *Let* $p_1, p_2 \in$ DIS. *Then* $\pi(p_1) = \pi(p_2) \;\;\Rightarrow\;\; \exists$ *pbf* $\mathbf{S}$ *s.t.* $p_1 S_\emptyset p_2$.

The proof of this proposition relies on the following definition and lemma.

Definition 3.8 For any $p_1, p_2 \in$ DIS such that $\pi(p_1) = \pi(p_2)$, let $\varphi(p_1, p_2)$ be the least relation on locations satisfying:

$$\begin{array}{l}
\varphi(nil, nil) \;=\; \varphi(x, x) \;=\; \varphi(\mu.\,r_1, \mu.\,r_2) \;=\; \varphi((r_1 + s_1)\,,\,(r_2 + s_2)) \;=\; \{(\varepsilon, \varepsilon)\} \\
\varphi(\,r_1\backslash\alpha,\, r_2\backslash\alpha\,) \;=\; \varphi(r_1\,\langle f\rangle,\, r_2\,\langle f\rangle) \;=\; \varphi(r_1, r_2) \\
\varphi(\,r_1[rec\,x.\; r_1/x],\, r_2[rec\,x.\; r_2/x]\,) \;\subseteq\; \varphi(\,rec\,x.\; r_1,\, rec\,x.\; r_2\,) \\
\varphi(\,(l_1 :: r_1 \mid k_1 :: s_1)\,,\; (l_2 :: r_2 \mid k_2 :: s_2)\,) \;=\; \begin{array}[t]{l} \{(\varepsilon, \varepsilon)\} \\ \cup\; (l_1, l_2)\cdot\varphi(r_1, r_2) \\ \cup\; (k_1, k_2)\cdot\varphi(s_1, s_2) \end{array}
\end{array}$$

where for any relation φ, we let $(l, l')\cdot\varphi =_{\rm def} \{(lu, l'v) \mid (u, v) \in \varphi\}$.

It may be easily checked that the relation $\varphi(p_1, p_2)$ is a consistent location association. Note that $\varphi(p_1, p_2)$ only relates those locations of p_1 and p_2 which are "statically exhibited", i.e. which do not occur under a dynamic operator. The following lemma establishes the relation between $\varphi(p_1, p_2)$ and the transitions of p_1, p_2.

Lemma 3.9 *Let* $p_1, p_2 \in$ DIS *be such that* $\pi(p_1) = \pi(p_2)$. *Then:*

1. $p_1 \overset{a}{\underset{u}{\Longrightarrow}}_s p_1' \;\;\Rightarrow\;\; \exists\, p_2', v$ *s.t.* $p_2 \overset{a}{\underset{v}{\Longrightarrow}}_s p_2'\,,\;\; \varphi(p_1, p_2) \cup (u, v) \;\subseteq\; \varphi(p_1', p_2')$ *and* $\pi(p_1') = \pi(p_2')$
2. $p_1 \overset{\tau}{\Longrightarrow}_s p_1' \;\;\Rightarrow\;\; \exists\, p_2'$ *s.t.* $p_2 \overset{\tau}{\Longrightarrow}_s p_2'\,,\;\; \varphi(p_1, p_2) \subseteq \varphi(p_1', p_2')$ *and* $\pi(p_1') = \pi(p_2')$

PROOF: By induction on the proofs of transitions of distributed processes. Note that in general $\varphi(p_1, p_2) \subseteq \varphi(p_1', p_2')$ because new parallel structure may appear as the computations proceed. □

We may now prove the above proposition.

Proof of Proposition 3.7: Define the family $\mathbf{T} = \{T_\varphi \mid \varphi \in \Phi\}$ by letting:

$$T_\varphi = \{ (r_1, r_2) \mid \pi(r_1) = \pi(r_2) \text{ and } \varphi \subseteq \varphi(r_1, r_2) \}$$

It is clear that $r_1 T_\emptyset r_2$ for any r_1, r_2 such that $\pi(r_1) = \pi(r_2)$. Let us show that $\mathbf{T}$ is a progressive bisimulation family. Suppose that $r_1 T_\varphi r_2$. If $r_1 \overset{a}{\underset{u}{\Longrightarrow}}_s r_1'$ then by Lemma 3.9 $r_2 \overset{a}{\underset{v}{\Longrightarrow}}_s r_2'$, with $\varphi(r_1, r_2) \cup (u, v) \subseteq \varphi(r_1', r_2')$ and $\pi(r_1') = \pi(r_2')$. We want to show that $\varphi' = \varphi \cup (u, v)$ is a *cla* and that $r_1' T_{\varphi'} r_2'$. But this follows immediately from $\varphi \subseteq \varphi(r_1, r_2)$ and Lemma 3.9, since $\varphi' = \varphi \cup (u, v) \subseteq \varphi(r_1, r_2) \cup (u, v) \subseteq \varphi(r_1', r_2')$. □

Using this proposition, we can finally prove that:

Proposition 3.10 *The relation $\approx_\ell^s$ is an equivalence on CCS processes.*

Proof: *Reflexivity:* Consider the family $\mathbf{S} = \{S_\varphi \mid \varphi \in \Phi\}$ defined by:

$$S_\varphi = \begin{cases} \{ (q, q) \mid q \in \text{DIS} \} & \text{if } \varphi \subseteq Id \\ \emptyset & \text{otherwise} \end{cases}$$

It is clear that $\mathbf{S}$ is a progressive bisimulation family such that $(q, q) \in S_\emptyset$ for any $q \in$ DIS. Hence $p \approx_\ell^s p$ for any $p \in$ CCS.

Symmetry: Let $p \approx_\ell^s q$. This means that for some $\bar{p}, \bar{q} \in$ DIS such that $\pi(\bar{p}) = p$ and $\pi(\bar{q}) = q$, there exists a progressive bisimulation family $\mathbf{S} = \{S_\varphi \mid \varphi \in \Phi\}$ such that $\bar{p}\, S_\emptyset\, \bar{q}$. Define now a family $\mathbf{R} = \{R_\psi \mid \psi \in \Phi\}$ by:

$$R_\psi = \{ (r, s) \mid s\, S_{\psi^{-1}}\, r \}$$

Clearly $\bar{q}\, R_\emptyset\, \bar{p}$. We show now that $\mathbf{R}$ is a progressive bisimulation family. Suppose $(r, s) \in R_\psi$: this is because $s\, S_\varphi\, r$, with $\varphi = \psi^{-1}$. Then $r \overset{a}{\underset{u}{\Longrightarrow}}_s r'$ implies $s \overset{a}{\underset{v}{\Longrightarrow}}_s s'$, with $s' S_{\varphi \cup \{(v,u)\}} r'$. Now $[\varphi \cup \{(v, u)\}]^{-1} = \psi^{-1} \cup \{(u, v)\}$, and thus $(r', s') \in R_{\psi^{-1} \cup \{(u,v)\}}$. The case of unobservable transitions is similar. We can then conclude that $q \approx_\ell^s p$.

Transitivity: Let $p \approx_\ell^s r$ and $r \approx_\ell^s q$. This means that for some $\bar{p}, \bar{q}, r_1, r_2 \in$ DIS such that $\pi(\bar{p}) = p$, $\pi(\bar{q}) = q$, $\pi(r_1) = \pi(r_2) = r$, there exist *pbf's* $\mathbf{R}^1$ and $\mathbf{R}^2$ s.t. $\bar{p} R_\emptyset^1 r_1$, $r_2 R_\emptyset^2 \bar{q}$. Moreover, if $\mathbf{T}$ is the *pbf* introduced in the proof of Proposition 3.7, we know (from the same proposition) that $r_1 T_\emptyset r_2$. Hence if we define a family $\mathbf{S} = \{S_\varphi \mid \varphi \in \Phi\}$ as follows:

$$\begin{aligned} (s, s') \in S_\varphi \;\Leftrightarrow\; & \exists\, t_1, t_2 \in \text{DIS such that } \pi(t_1) = \pi(t_2); \\ & \exists\, pbf's\ \mathbf{S}^1, \mathbf{S}^2,\ \exists\, cla's\ \varphi_1, \psi, \varphi_2, \text{ such that} \\ & \varphi \subseteq \varphi_2 \circ \psi \circ \varphi_1 \text{ and } s\, S_{\varphi_1}^1\, t_1\, T_\psi\, t_2\, S_{\varphi_2}^2\, s' \end{aligned}$$

it is clear that $\bar{p}\, S_\emptyset\, \bar{q}$. Furthermore it is easy to check that $\mathbf{S}$ is a progressive bisimulation family, since for any u, v, v', w we have $\varphi \cup \{(u,w)\} \subseteq \varphi_2 \cup \{(v',w)\} \circ \psi \cup \{(v,v')\} \circ \varphi_1 \cup \{(u,v)\}$. □

A pleasant consequence of Proposition 3.7 is that $\approx^s_\ell$ is independent from the particular distributions we choose. If two CCS terms p and q are equivalent, then any two distributions of them are related by $S_\emptyset$, for some progressive bisimulation family $\mathbf{S} = \{S_\varphi \mid \varphi \in \Phi\}$.

Corollary 3.11 *For any* $p, q \in \mathrm{CCS}$ *:* $p \approx^s_\ell q \;\Leftrightarrow\;$ *for all* $\bar{p}, \bar{q} \in \mathrm{DIS}$ *such that* $p\,\mathcal{D}\,\bar{p}$ *and* $q\,\mathcal{D}\,\bar{q}$ *there exists a progressive bisimulation family* $\mathbf{S} = \{S_\varphi \mid \varphi \in \Phi\}$ *such that* $\bar{p}\, S_\emptyset\, \bar{q}$*.*

By virtue of this result, we can restrict attention to particular "binary" distributions, systematically associating location 0 to the left operand and location 1 to the right operand of a parallel composition. A distribution of this kind will be called *canonical.* Similarly, elements of $\{0,1\}^*$ will be called *canonical locations.* These are exactly the locations used in [MN92] and, with a slightly different notation, in [MY92],[Yan93]. In fact, when applied to canonical distributions of CCS terms, our transition rules give exactly the same transitions as the *spatial transition system* of Montanari and Yankelevich (except for τ-transitions, for which they use pairs of locations).

Let us see now a simple example, which shows the difference between location equivalence and causality-based equivalences, such as the (weak) causal bisimulation of [DD90]:

Example 3.12 $\quad a.\,b + b.\,a \quad \not\approx^s_\ell \quad (a.\,\gamma \mid \bar{\gamma}.\,b)\backslash\gamma \;+\; (b.\,\gamma \mid \bar{\gamma}.\,a)\backslash\gamma \quad \approx^s_\ell \quad a \mid b$

Using canonical distributions, it is easy to see that the computation a followed by b yields the association $\varphi = \{\,(\varepsilon,0)\,,(\varepsilon,1)\,\}$ between the locations of the first two processes, which is *not* consistent. On the other hand, for the second and third process we build the consistent association $\varphi_1 = \{\,(0,0)\,,(1,1)\,\}$ for the computation a followed by b, and the consistent association $\varphi_2 = \{\,(0,1)\,,(1,0)\,\}$ for the computation b followed by a.

Another example, showing the difference w.r.t. the usual CCS semantics, is:

Example 3.13 $\quad rec\,x.\; a.\,x \quad \not\approx^s_\ell \quad (rec\,x.\; a.\,x \mid rec\,x.\; a.\,x)$

In the standard CCS semantics these two processes give rise to isomorphic transition systems.

3.2 Static location preorder

We define now a preorder $\sqsubseteq_\ell^s$ on CCS processes, which formalises the idea that one process is more sequential or *less distributed* than another. This preorder is obtained by slightly relaxing the notion of consistent association. The intuition for $p \sqsubseteq_\ell^s q$ is that there exist two distributions $\bar{p}$ and $\bar{q}$ of them such that whenever $\bar{p}$ can perform two transitions at independent locations, then $\bar{q}$ performs corresponding transitions at locations which are also independent, while the reverse is not necessarily true. This is expressed by the following notion of left-consistency:

Definition 3.14 A relation $\varphi \subseteq (Loc^* \times Loc^*)$ is a *left-consistent location association* if:

$$(u,v) \in \varphi \quad \& \quad (u',v') \in \varphi \quad \Rightarrow \quad (u \diamond u' \;\; \Rightarrow \;\; v \diamond v')$$

Now, if Ψ is the set of left-consistent location associations, we may obtain a notion of *progressive pre-bisimulation family (ppbf)* on distributed processes of DIS by simply replacing Φ by Ψ in Definition 3.5. Again, this gives rise to a relation on CCS processes:

Definition 3.15 (**Static location preorder**) If $p, q \in$ CCS, let $p \sqsubseteq_\ell^s q$ if and only if for some $\bar{p}, \bar{q} \in$ DIS such that $p\,\mathcal{D}\,\bar{p}$ and $q\,\mathcal{D}\,\bar{q}$, there exists a progressive pre-bisimulation family $\mathbf{S} = \{S_\psi \mid \psi \in \Psi\}$ such that $\bar{p}\, S_\emptyset\, \bar{q}$.

It is easy to see that $p \approx_\ell^s q \;\Rightarrow\; p \sqsubseteq_\ell^s q$. As may be expected the reverse is not true. We have for instance, resuming the examples of the previous section:

Example 3.16 $\quad a.\,b + b.\,a \quad \sqsubseteq_\ell^s \quad (a.\,\gamma \mid \bar{\gamma}.\,b)\backslash\gamma \;+\; (b.\,\gamma \mid \bar{\gamma}.\,a)\backslash\gamma$

Example 3.17 $\quad rec\,x.\;a.\,x \quad \sqsubseteq_\ell^s \quad (rec\,x.\;a.\,x \mid rec\,x.\;a.\,x)$

Having introduced both an equivalence $\approx_\ell^s$ and a preorder $\sqsubseteq_\ell^s$ based on the same intuition, we may wonder whether $\approx_\ell^s$ coincides with the equivalence $\simeq_\ell^s =_{def} \sqsubseteq_\ell^s \cap \sqsupseteq_\ell^s$ induced by the preorder. It is clear that $\approx_\ell^s \subseteq \simeq_\ell^s$, since we have both $\approx_\ell^s \subseteq \sqsubseteq_\ell^s$ and $\approx_\ell^s \subseteq \sqsupseteq_\ell^s$. On the other hand, the kernel of the preorder is weaker than location equivalence, as shown by the following example. Consider the two processes:

$$a.\,a.\,a \;+\; (a \mid a \mid a) \qquad \text{and} \qquad a.\,a.\,a \;+\; a.\,a \mid a \;+\; (a \mid a \mid a)$$

These two processes are not equivalent w.r.t. $\approx_\ell^s$, but they are equivalent w.r.t. $\simeq_\ell^s$ because $a.\,a.\,a \sqsubseteq_\ell^s a.\,a \mid a \sqsubseteq_\ell^s a \mid a \mid a$.

We will show in Section 5 that the static preorder $\sqsubseteq_\ell^s$ coincides with the dynamic location preorder $\sqsubseteq_\ell^d$ of [BCHK91b], and thus inherits the theory of the latter.

4 Dynamic approach

We briefly recall here the dynamic approach of [BCHK91], and in particular the definitions of $\approx_\ell^d$ and $\sqsubseteq_\ell^d$. In the dynamic approach, locations are associated with actions rather than with parallel components. This association is built dynamically, according to the rule:

$$\text{(D1)} \qquad a.p \xrightarrow[l]{a}_d l :: p \qquad \text{for any } l \in Loc$$

In some sense locations are transmitted from transitions to processes, whereas in the static case we had the inverse situation. Rule (D1) is the essence of the dynamic location semantics. The remaining rules for observable transitions are just as in the static semantics, see Figure 2 at p. 9. We refer to [BCHK91] for more intuition on the dynamic notion of location. Let us just observe that these locations increase at each step, even if the execution goes on within the same parallel component. In fact the location l which appears in rule (D1) may be seen as an identifier for action a, or more precisely, for that particular occurrence of a. Then the location u of a generic transition $p \xrightarrow[u]{a}_d p'$ is a record of all the action occurrences which causally precede a (through the prefixing operator), what we shall call also the *access path* to a.

The observable dynamic transitions $p \xrightarrow[u]{a}_d p'$ are related to the static transitions $p \xrightarrow[u]{a}_s p'$ in a simple way. To see this, let us introduce a few notations. Let $\Delta_k(p)$ be the function that erases the atomic location k in p, wherever it occurs. Formally, $\Delta : (Loc \times \text{LCCS}) \rightarrow \text{LCCS}$ is defined by:

$$\begin{array}{lcl} \Delta_k(p) & = & p, \quad \text{if } p \in \text{CCS} \\ \Delta_k(l :: p) & = & \begin{cases} \Delta_k(p) & \text{if } k = l \\ l :: \Delta_k(p) & \text{otherwise} \end{cases} \end{array}$$

as well as clauses stating the compatibility of Δ_k with the remaining operators (for instance $\Delta_k(p \mid q) = \Delta_k(p) \mid \Delta_k(q)$, etc.). Also, for any $p \in \text{LCCS}$, let $Loc(p)$ be the set of atomic locations occurring in p, defined in the obvious way. We have then the following correspondence between the two kinds of location transitions:

Fact 4.1 *Let $p \in LCCS$, $l \notin Loc(p)$. Then:*

$$\begin{array}{llcl} (i) & p \xrightarrow[ul]{a}_d p' & \Rightarrow & p \xrightarrow[u]{a}_s \Delta_l(p') \\ (ii) & p \xrightarrow[u]{a}_s p' & \Rightarrow & \exists\, p'' \text{ s.t. } p \xrightarrow[ul]{a}_d p'' \text{ and } \Delta_l(p'') = p' \end{array}$$

The proof, by induction on the inference of transitions, is left to the reader. □

Because of rule (D1), the dynamic location transition system is both infinitely branching and acyclic: it thus gives infinite representations for all regular processes. Indeed, this has been the main criticism addressed to the dynamic approach, see

[Ace91],[MY92],[MN92],[Mur93]. In fact, while the infinite branching may be overcome easily (through a *canonical* choice of dynamic locations, see Appendix), the infinite progression is really intrinsic to the dynamic semantics.

Note that for τ-transitions, for which we do not want to introduce additional locations, we simply use the static transition rules. Although this last point differentiates our strong dynamic location transition system from that originally introduced in [BCHK91], where no locations were associated with τ-transitions, the resulting *weak (dynamic) location transition system* is the same. The definition of the weak transitions $\underset{u}{\overset{a}{\Longrightarrow}}_d$ and $\overset{\tau}{\Longrightarrow}_d$ is similar to that of the $\underset{u}{\overset{a}{\Longrightarrow}}_s$ and $\overset{\tau}{\Longrightarrow}_s$.

We define now the *dynamic location equivalence* $\approx_\ell^d$ and the *dynamic location preorder* $\sqsubseteq\!\!\!\!\sim_\ell^d$. Because of the flexibility in the choice of locations, these definitions are much simpler than in the static case. In [BCHK91] the relations $\approx_\ell^d$ and $\sqsubseteq\!\!\!\!\sim_\ell^d$ are obtained as instances of a general notion of *parameterized location bisimulation*. We shall use here directly the instantiated definitions.

Definition 4.2 (**Dynamic location equivalence**) A relation $R \subseteq \mathrm{LCCS} \times \mathrm{LCCS}$ is called a *dynamic location bisimulation (dlb)* iff for all $(p, q) \in R$ and for all $a \in Act, u \in Loc^+$:

(1) $p \underset{u}{\overset{a}{\Longrightarrow}}_d p' \;\Rightarrow\; \exists\, q'$ such that $q \underset{u}{\overset{a}{\Longrightarrow}}_d q'$ and $(p', q') \in R$

(2) $q \underset{u}{\overset{a}{\Longrightarrow}}_d q' \;\Rightarrow\; \exists\, p'$ such that $p \underset{u}{\overset{a}{\Longrightarrow}}_d p'$ and $(p', q') \in R$

(3) $p \overset{\tau}{\Longrightarrow}_d p' \;\Rightarrow\; \exists\, q'$ such that $q \overset{\tau}{\Longrightarrow}_d q'$ and $(p', q') \in R$

(4) $q \overset{\tau}{\Longrightarrow}_d q' \;\Rightarrow\; \exists\, p'$ such that $p \overset{\tau}{\Longrightarrow}_d p'$ and $(p', q') \in R$

The largest *dlb* is called *dynamic location equivalence* and denoted $\approx_\ell^d$.

We refer to [BCHK91] for examples and results concerning $\approx_\ell^d$. Consider now the location preorder $\sqsubseteq\!\!\!\!\sim_\ell^d$. Here, instead of requiring the identity of locations in corresponding transitions, we demand that the locations in the second (more distributed) process be subwords of the locations in the first (more sequential) process. Formally, the *subword* relation $\leq_{\mathrm{sub}}$ on Loc^* is defined by: $v \leq_{\mathrm{sub}} u \;\Leftrightarrow\; \exists\, v_1, \ldots, v_k\,,\ \exists\, w_1, \ldots, w_{k+1}$ s.t. $v = v_1 \cdots v_k$ and $u = w_1 v_1 \cdots w_k v_k w_{k+1}$.

Definition 4.3 (**Dynamic location preorder**) A relation $R \subseteq \mathrm{LCCS} \times \mathrm{LCCS}$ is called a *dynamic location pre-bisimulation (dlpb)* iff for all $(p, q) \in R$ and for all $a \in Act, u \in Loc^+$:

(1) $p \underset{u}{\overset{a}{\Longrightarrow}}_d p' \;\Rightarrow\; \exists\, v \leq_{\mathrm{sub}} u, \exists\, q'$ such that $q \underset{v}{\overset{a}{\Longrightarrow}}_d q'$ and $(p', q') \in R$

(2) $q \underset{v}{\overset{a}{\Longrightarrow}}_d q' \;\Rightarrow\; \exists\, u.\ v \leq_{\mathrm{sub}} u, \exists\, p'$ such that $p \underset{u}{\overset{a}{\Longrightarrow}}_d p'$ and $(p', q') \in R$

(3) $p \overset{\tau}{\Longrightarrow}_d p' \;\Rightarrow\; \exists\, q'$ such that $q \overset{\tau}{\Longrightarrow}_d q'$ and $(p', q') \in R$

(4) $q \overset{\tau}{\Longrightarrow}_d q' \;\Rightarrow\; \exists\, p'$ such that $p \overset{\tau}{\Longrightarrow}_d p'$ and $(p', q') \in R$

The largest *dlpb* is called *dynamic location preorder* and denoted $\sqsubseteq\!\!\!\!\sim_\ell^d$.

The intuition is as follows. If p is a sequentialized version of q, then each component of p corresponds to a group of parallel components in q. Thus the local causes of any action of q will correspond to a subset of local causes of the corresponding action of p. This may be easily verified for the following examples:

Example 4.4 $\quad a.\,a.\,a \;\; \sqsubseteq\!\!\!\!\sim_{\ell}^{d} \;\; a.\,a \mid a \qquad$ and $\qquad a.\,b \,+\, b.\,a \;\; \sqsubseteq\!\!\!\!\sim_{\ell}^{d} \;\; a \mid b$

We shall not comment further here on the relations $\approx_{\ell}^{d}$ and $\sqsubseteq\!\!\!\!\sim_{\ell}^{d}$, referring again the reader to [BCHK91] for more examples and for results concerning these relations. We proceed instead to state our main result, namely that the dynamic relations $\approx_{\ell}^{d}$ and $\sqsubseteq\!\!\!\!\sim_{\ell}^{d}$ coincide with the static relations $\approx_{\ell}^{s}$ and $\sqsubseteq\!\!\!\!\sim_{\ell}^{s}$ introduced in the previous section.

Theorem 4.5 *Let $p, q \in$ CCS. Then:*

$$(\mathbf{1}) \quad p \approx_{\ell}^{s} q \quad \Leftrightarrow \quad p \approx_{\ell}^{d} q$$
$$(\mathbf{2}) \quad p \sqsubseteq\!\!\!\!\sim_{\ell}^{s} q \quad \Leftrightarrow \quad p \sqsubseteq\!\!\!\!\sim_{\ell}^{d} q$$

The rest of the paper is devoted to proving this theorem. To this end, we shall use a new transition system on CCS, called *occurrence transition system*, which in some sense incorporates the information of both location transition systems. This system will serve as an intermediate between the static and the dynamic semantics. The main point will be to prove that starting from a static or a dynamic location computation, one may always reconstruct a corresponding occurrence computation. This means, essentially, that all the information about distribution and local causality is already present in both location transition systems.

The two location transition systems could also be compared directly, without recourse to an auxiliary transition system. However we find it instructive to introduce the occurrence transition system, since it provides a concrete level of description where the notions of *occurrence* of an action, access path to an occurrence and *computation state* have a precise definition. Moreover, as we shall see, it allows for the definition of a notion of *local history preserving bisimulation*, which turns out to be a third equivalent formulation of location equivalence.

5 Equivalence of the two approaches

5.1 The occurrence transition system

To compare the two location semantics we introduce a new transition system, called *occurrence (transition) system*, whose states represent CCS computation states with a “past”, and whose labels are occurrences of actions within a computation. This system, which is based on a syntactic notion of *occurrence* of action, is essentially a simplification of the *event (transition) system* introduced in [BC91] to compare different models of CCS: it is simpler because we do not try to identify uniquely all occurrences of action in a term, as in [BC91], but only those which can coexist in a computation. Moreover, since we are interested here in weak semantics, we shall not distinguish between τ-actions and we concentrate on *abstract* occurrences, in which τ-actions and communications are absorbed. Formally, the set $\mathcal{O}_\tau$ of *occurrences* is defined as $\mathcal{O}_\tau = \mathcal{O} \cup \{\tau\}$, where $\mathcal{O}$, the set of visible occurrences, is given by the grammar:

$$e \ ::= \ a \ \mid \ a\,e \ \mid \ 0\,e \ \mid \ 1\,e$$

The meaning of the occurrence constructors is as follows: a denotes an initial occurrence of action a (possibly following - or followed by - some τ actions), $a\,e$ denotes the occurrence e after an action a, while $0\,e, 1\,e$ represent the occurrence e at the left, resp. at the right of a parallel operator. Finally the symbol τ is used - with abuse of notation - to represent any occurrence of a τ-action in a computation. We use $e, e', \ldots$ to range over the whole set $\mathcal{O}_\tau$.

We show now that a visible occurrence $e \in \mathcal{O}$ incorporates both its static and its dynamic location. For note that $\mathcal{O}$ could also be defined as:

$$\mathcal{O} = (Act \cup \{0,1\})^*\, Act$$

Then an occurrence $e \in \mathcal{O}_\tau$ is either τ or a word σa, for some $\sigma \in (Act \cup \{0,1\})^*$ and $a \in Act$. The *label* of $e \in \mathcal{O}_\tau$ is the action of which e is an occurrence:

Definition 5.1 (**Label**) The function $\lambda : \mathcal{O}_\tau \rightarrow Act_\tau$ is defined by:

$$\lambda(\tau) = \tau, \ \lambda(\sigma a) = a.$$

This alternative presentation of $\mathcal{O}$ makes it also particularly easy to define the *location* and the *access path* of a visible occurrence e. The location $loc(e)$ of an occurrence $e \in \mathcal{O}$ is its projection on the set of canonical locations:

Definition 5.2 (**Location**) The function $loc : \mathcal{O} \rightarrow \{0,1\}^*$ is defined by:

$$loc(e) = e \upharpoonright \{0,1\}^*$$

The relation of causality on visible occurrences is simply the prefix ordering on $\mathcal{O}$.

Definition 5.3 (**Local causality**) The relation $\preceq \subseteq (\mathcal{O} \times \mathcal{O})$ is given by:

$$e \preceq e' \ \Leftrightarrow \ e = e' \ \text{ or } \ \exists\, e'' \ \text{ s.t. } \ ee'' = e'$$

We know that $\preceq$ is a partial ordering. We call $\preceq$ *local causality* because it only connects occurrences within the same parallel component: it is clear that $e \preceq e' \Rightarrow loc(e) \ll loc(e')$. Let as usual $e \prec e'$ stand for $(e \preceq e' \;\&\; e \neq e')$. For $e \in \mathcal{O}$, we define $\downarrow\! e = \{e' \mid e' \prec e\}$ to be the set of *local causes* of e. Then the access path of e is the sequence of such causes:

Definition 5.4 (**Access path**) For $e \in \mathcal{O}$, $path(e) =_{\text{def}} e_1 \cdot \;\cdots\; \cdot e_n$, where $\{e_1, \ldots, e_n\} = \downarrow\! e$ and $e_i \prec e_{i+1}$, $1 \leq i < n$.

For instance, if $e = 0a10b11c$, then $\downarrow\! e = \{0a, 0a10b\}$ and $path(e) = (0a) \cdot (0a10b)$. We call $e \in \mathcal{O}$ an *initial occurrence* if $\downarrow\! e = \emptyset$ (equivalently $path(e) = \varepsilon$). An initial occurrence has always the form $e = loc(e) \cdot \lambda(e)$. More generally, if $\eta' \ll \eta$ and η/η' is the residual of η after η', defined by $\eta/\eta' = \eta''$ if $\eta = \eta'\eta''$, we have the following characterisation for visible occurrences:

Fact 5.5 *An occurrence $e \in \mathcal{O}$ is completely determined by its label, location and access path. Namely, if $\lambda(e) = a$, $loc(e) = \eta$ and $path(e) = e_1 \cdot \;\cdots\; \cdot e_n$, $n \geq 1$, then:*

$$e = (loc(e_1) \cdot \lambda(e_1)) \cdot (loc(e_2)/loc(e_1) \cdot \lambda(e_2)) \cdot \;\cdots\; \cdot (\eta/loc(e_n) \cdot a)$$

If $path(e) = \varepsilon$ then $e = \eta \cdot a$.

We define now the relation of *concurrency* on visible occurrences:

Definition 5.6 (**Concurrency**) The relation $\smile \subseteq (\mathcal{O} \times \mathcal{O})$ is defined by:

$$e \smile e' \quad \Leftrightarrow \quad \begin{cases} \text{either} & e = \sigma\, 0\, e_0 \quad \& \quad e' = \sigma\, 1\, e'_0 \\ \text{or} & e = \sigma\, 1\, e_0 \quad \& \quad e' = \sigma\, 0\, e'_0 \end{cases}$$

where $\sigma \in (Act \cup \{0,1\})^*$, $e_0, e'_0 \in \mathcal{O}$.

The relation $\smile$ is symmetric and irreflexive. Clearly $e \smile e' \;\Rightarrow\; loc(e) \diamond loc(e')$.

Let us now shift attention to the states of the occurrence system. As we said, these states are meant to represent processes with a *past*. The past records the observable guards which have been passed along a computation. Formally, if $p, q \in$ CCS, the set $\mathcal{S}$ of *computation states* is given by:

$$\xi \;::=\; nil \;\mid\; \mu.p \;\mid\; p+q \;\mid\; x \;\mid\; rec\,x.\,p \;\mid\; \widehat{a}.\,\xi \;\mid\; (\xi \mid \xi') \;\mid\; \xi\backslash\alpha \;\mid\; \xi\,\langle f\rangle$$

The construct $\widehat{a}.\,\xi$ is used to represent the state ξ with "past" a, that is, after a guard a has been passed. The idea is that any transition labelled by a visible occurrence will introduce a "hat" in the resulting state. The basic operational rule is:

$$\text{(O1)} \qquad a.\,p \xrightarrow{a} \widehat{a}.\,p$$

On the other hand an invisible occurrence τ does not leave any trace in the past. This is expressed by the rule:

(O1′) $\tau.p \xrightarrow{\tau} p$

The hats recorded in states are used to build up "deep" occurrences along computations, according to the rule:

(O2) $\xi \xrightarrow{e} \xi'\,,\quad e \neq \tau \quad\Rightarrow\quad \widehat{a}.\xi \xrightarrow{a\,e} \widehat{a}.\xi'$

On the other hand occurrences of the form $i\,e$, for $i = 0, 1$, originate from parallel terms $\xi \mid \xi'$:

(O3) $\xi \xrightarrow{e} \xi'\,,\quad e \neq \tau \quad\Rightarrow\quad \xi \mid \xi'' \xrightarrow{0\,e} \xi' \mid \xi''\,,\quad \xi'' \mid \xi \xrightarrow{1\,e} \xi'' \mid \xi'$

For defining the whole occurrence system we need a few more notations. First, we extend a relabelling f of actions to occurrences by letting: $f(a\,e) = f(a)\,f(e)$ and $f(i\,e) = i\,f(e),\ i = 0, 1$.

Moreover, we need an auxiliary function for defining the communication rule. In the occurrence system communication arises from concurrent occurrences with complementary labels. However the resulting τ-occurrence should not contribute to the past, since this only keeps track of observable actions. Thus we need to take back the hats introduced by the occurrences participating in the communication. To this end we introduce a function $\delta_e(\xi)$, which erases the hat corresponding to occurrence e in ξ (somehow similar to the function $\Delta_k(p)$ used in Section 3). The partial function $\delta : (\mathcal{O} \times \mathcal{S}) \rightarrow \mathcal{S}$ is given by:

$$\begin{array}{lcl}
\delta_a(\widehat{a}.p) & = & p \\
\delta_{ae}(\widehat{a}.\xi) & = & \widehat{a}.\delta_e(\xi) \\
\delta_{0e}(\xi \mid \xi') & = & \delta_e(\xi) \mid \xi' \\
\delta_{1e}(\xi \mid \xi') & = & \xi \mid \delta_e(\xi') \\
\delta_e(\xi \backslash \alpha) & = & \delta_e(\xi)\backslash\alpha \\
\delta_e(\xi\,\langle f\rangle) & = & \delta_e(\xi)\,\langle f\rangle
\end{array}$$

We have now all the elements to define the occurrence system for CCS. The rules specifying this system are listed in Figure 3. Note that the condition $\lambda(e) \neq \{\alpha, \bar{\alpha}\}$ in rule (O6) could be strenghtened to $e \upharpoonright \{\alpha, \bar{\alpha}\} = \varepsilon$, to prevent transitions like $\widehat{a}\,.\,b.p \xrightarrow{a\,b} \widehat{a}\,.\,\widehat{b}.p$. However this would make no difference for states ξ obtained via an occurrence computation from a CCS term (more will be said on this point below). The *weak occurrence system* is now given by:

$$\begin{array}{lcl}
\xi \overset{\tau}{\Longrightarrow} \xi' & \Leftrightarrow & \exists\,\xi_0,\ldots,\xi_n \ \text{ s.t. }\ \xi = \xi_0 \xrightarrow{\tau} \xi_1 \cdots \xrightarrow{\tau} \xi_n = \xi' \\
\xi \overset{e}{\Longrightarrow} \xi' & \Leftrightarrow & \exists\,\xi_1,\xi_2 \ \ s.t.\ \ \xi \overset{\tau}{\Longrightarrow} \xi_1 \xrightarrow{e} \xi_2 \overset{\tau}{\Longrightarrow} \xi'
\end{array}$$

Let us examine some properties of this weak occurrence system. It is clear that any term ξ gives an intensional representation of a CCS computation state. In fact from each state ξ one may extract the set of visible occurrences that have led to it. Clearly this set should be empty for a CCS term. Formally, the set of *past occurrences* of a term ξ is defined by:

For each $e \in \mathcal{O}_\tau$ let $\xrightarrow{e} \subseteq (\mathcal{S} \times \mathcal{S})$ be the least binary relation satisfying the following axioms and rules.

(O1) $a.p \xrightarrow{a} \widehat{a}.\,p$

(O1′) $\tau.p \xrightarrow{\tau} p$

(O2) $\xi \xrightarrow{e} \xi' ,\quad e \neq \tau \quad \Rightarrow \quad \widehat{a}.\,\xi \xrightarrow{a\,e} \widehat{a}.\,\xi'$

(O3) $\xi \xrightarrow{e} \xi' ,\quad e \neq \tau \quad \Rightarrow \quad \xi \mid \xi'' \xrightarrow{0\,e} \xi' \mid \xi''$

$\xi'' \mid \xi \xrightarrow{1\,e} \xi'' \mid \xi'$

(O2′) $\xi \xrightarrow{\tau} \xi' \quad \Rightarrow \quad \widehat{a}.\,\xi \xrightarrow{\tau} \widehat{a}.\,\xi'$

(O3′) $\xi \xrightarrow{\tau} \xi' \quad \Rightarrow \quad \xi \mid \xi'' \xrightarrow{\tau} \xi' \mid \xi''$

$\xi'' \mid \xi \xrightarrow{\tau} \xi'' \mid \xi'$

(O4) $\left.\begin{array}{l} \xi_0 \xrightarrow{e_0} \xi_0' \ , \quad \xi_1 \xrightarrow{e_1} \xi_1' \\ \text{and} \quad \lambda(e_0) = \overline{\lambda(e_1)} \end{array}\right\} \quad \Rightarrow \quad \xi_0 \mid \xi_1 \xrightarrow{\tau} \delta_{e_0}(\xi_0') \mid \delta_{e_1}(\xi_1')$

(O5) $p \xrightarrow{e} \xi \quad \Rightarrow \quad p + q \xrightarrow{e} \xi$

$q + p \xrightarrow{e} \xi$

(O6) $\xi \xrightarrow{e} \xi' ,\quad \lambda(e) \notin \{\alpha, \bar{\alpha}\} \quad \Rightarrow \quad \xi \backslash \alpha \xrightarrow{e} \xi' \backslash \alpha$

(O7) $\xi \xrightarrow{e} \xi' \quad \Rightarrow \quad \xi \, \langle f \rangle \xrightarrow{f(e)} \xi' \, \langle f \rangle$

(O8) $p\,[rec\,x.\ p/x] \xrightarrow{e} \xi \quad \Rightarrow \quad rec\,x.\ p \xrightarrow{e} \xi$

Figure 3: Occurrence transition system

$$\begin{array}{lcl}
occ\,(p) & = & \emptyset, \quad \text{if } p \in \text{CCS} \\
occ\,(\widehat{a}.\,\xi) & = & \{a\} \cup a \cdot occ\,(\xi) \\
occ\,(\xi \mid \xi') & = & 0 \cdot occ\,(\xi) \cup 1 \cdot occ\,(\xi') \\
occ\,(\xi \backslash \alpha) & = & \{\, e \in occ\,(\xi) \mid \lambda(e) \neq \alpha, \bar{\alpha}\,\} \\
occ\,(\xi\,\langle f \rangle) & = & occ\,(\xi)
\end{array}$$

For states ξ reachable from a CCS term, the clause for restriction reduces to $occ\,(\xi\backslash\alpha) = occ\,(\xi)$. In fact, one may easily verify that such states, which we shall call CCS *computation states*, are exactly those ξ whose subterms $\xi'\backslash\alpha$ satisfy $\alpha, \bar{\alpha} \notin \lambda(occ\,(\xi'))$. As may be expected, we have the following:

Remark 5.7 *If $\delta_e(\xi)$ is defined, then $occ\,(\delta_e(\xi)) = occ\,(\xi) - \{e\}$.*

The next lemma shows how visible occurrences are generated along computations, and how they are related.

Lemma 5.8 *Let ξ be a* CCS *computation state. Then:*

$$1.\quad \xi \xrightarrow{e} \xi',\ e \neq \tau \quad \Rightarrow \quad \begin{cases} (i)\ \ occ\,(\xi') = occ\,(\xi) \cup \{e\} \\ (ii)\ \ \forall e' \in occ\,(\xi):\ e' \prec e \ \text{ or } \ e' \smile e\,) \\ (iii)\ \ {\downarrow} e \subseteq occ\,(\xi) \end{cases}$$

$$2.\quad \xi \xrightarrow{\tau} \xi' \quad \Rightarrow \quad occ\,(\xi') = occ\,(\xi)$$

Proof: We start by showing *1.*, by induction on the proof of $\xi \xrightarrow{e} \xi',\ e \neq \tau$. Note that in the cases where ξ is a CCS term, namely when the last rule applied is (O1), (O5) or (O8), $occ\,(\xi) = \emptyset$ and ${\downarrow}e = \emptyset$, thus conditions (ii) and (iii) are trivially satisfied. We consider some representative cases.

– (O1) $\xi = ap \xrightarrow{a} \widehat{a}\,.p = \xi'$. Then (i) is straightforward since $occ\,(a.p) = \emptyset$ and $occ\,(\,\widehat{a}\,.p) = \{a\}$.

– (O2) $\xi = \widehat{a}\,.\xi_0 \xrightarrow{a\,e_0} \widehat{a}\,.\xi_0' = \xi'$ is inferred from $\xi_0 \xrightarrow{e_0} \xi_0'$. Let us first check condition (i). By induction $occ\,(\xi_0') = occ\,(\xi_0) \cup \{e_0\}$. Then $occ\,(\xi') = \{a\} \cup a\,(occ\,(\xi_0)) \cup \{a\,e_0\} = occ\,(\xi) \cup \{a\,e_0\}$. Consider now condition (ii). By induction, we have: $\forall e_0' \in occ\,(\xi_0):\ e_0' \prec e_0$ or $e_0' \smile e_0$. Now $occ\,(\xi) = \{a\} \cup a\,(occ\,(\xi_0))$. Clearly $a \prec a\,e_0$, whereas for $e' = a\,e_0'$, $e_0' \in occ\,(\xi_0)$, (ii) follows from induction and the fact that $\prec$ and $\smile$ are preserved by prefixing with a. As for (iii), ${\downarrow}(a\,e_0) \subseteq occ\,(\,\widehat{a}\,.\xi_0)$ follows from ${\downarrow}e_0 \subseteq occ\,(\xi_0)$.

– (O6) $\xi = \xi_0\backslash\alpha \xrightarrow{e} \xi_0'\backslash\alpha = \xi'$ is deduced from $\xi_0 \xrightarrow{e} \xi_0'$, $\lambda(e) \neq \alpha, \bar{\alpha}$. By induction $occ\,(\xi_0') = occ\,(\xi_0) \cup \{e_0\}$. Now $\alpha, \bar{\alpha} \notin \lambda(occ\,(\xi'))$ since $\xi' = \xi_0'\backslash\alpha$ is a

CCS computation state. Thus $occ\,(\xi_0'\backslash\alpha) = occ\,(\xi_0') = occ\,(\xi_0) \cup \{e\} = occ\,(\xi_0\backslash\alpha) \cup \{e\}$. Conditions (ii) and (iii) follow trivially by induction.

Point *2.* is proved by induction on the proof of the transition $\xi \xrightarrow{\tau} \xi'$. For the cases (O1$'$),(O5),(O8), it is enough to remark that, if $\xi \in$ CCS, then also $\xi' \in$ CCS, and thus $occ\,(\xi') = \emptyset = occ\,(\xi)$. We examine here the case where the last rule applied is (O4). The other cases are similar to the corresponding cases in point *1.*.

– (O4) $\xi = \xi_0 \mid \xi_1 \xrightarrow{\tau} \xi_0' \mid \xi_1' = \xi'$ is deduced from $\xi_0 \xrightarrow{e_0} \xi_0''$, $\xi_1 \xrightarrow{e_1} \xi_1''$, $\lambda(e_0) = \overline{\lambda(e_1)}$, and $\xi_i' = \delta_{e_i}(\xi_i'')$. By point *1.* $occ\,(\xi_i'') = occ\,(\xi_i) \cup \{e_i\}$, $i = 0, 1$. Now by the Remark 5.1, $occ\,(\delta_{e_i}(\xi_i'')) = occ\,(\xi_i'') - \{e_i\} = occ\,(\xi_i)$. Hence $occ\,(\xi') = occ\,(\xi)$. □

Corollary 5.9 *Let $p \in$ CCS. If $p \overset{e_1}{\Longrightarrow} \xi_1 \cdots \overset{e_n}{\Longrightarrow} \xi_n$, where $\forall i:\ e_i \neq \tau$, then:*

1. $\forall i:\ occ\,(\xi_i) = \{\, e_1, \ldots, e_i \,\}$
2. $i < j \ \Rightarrow\ e_i \prec e_j$ or $e_i \smile e_j$

Proof: Point *1.* is straightforward. As for *2.*, we have $\xi_{j-1} \overset{\tau}{\Longrightarrow} \xi' \xrightarrow{e_j} \xi'' \overset{\tau}{\Longrightarrow} \xi_j$ for some ξ', ξ''. Now $i < j \ \Rightarrow\ e_i \in occ\,(\xi_{j-1})$ by *1.*, whence the result by Lemma 5.8. □

The following proposition shows how local causality may be recovered from static locations along a computation:

Proposition 5.10 *Let $p \in CCS$. If $p \overset{e_1}{\Longrightarrow} \xi_1 \cdots \overset{e_n}{\Longrightarrow} \xi_n$, where $\forall i:\ e_i \neq \tau$, then:*

$$e_i \prec e_j \quad \Longleftrightarrow \quad i < j \ \text{ and } \ loc(e_i) \ll loc(e_j)$$

Proof: $\Leftarrow$: Since $i < j$, by Corollary 5.9 either $e_i \prec e_j$ or $e_i \smile e_j$. But it cannot be $e_i \smile e_j$, since this would imply $loc(e_i) \diamond loc(e_j)$. Thus $e_i \prec e_j$.
$\Rightarrow$: If $e_i \prec e_j$, then it cannot be $j < i$, because of Corollary 5.9 again. Moreover, since e_i is a prefix of e_j, also $loc(e_i)$ is a prefix of $loc(e_j)$. □

We proceed to define a notion of bisimulation on the weak occurrence system. Once again we use a notion of consistency and progressive bisimulation family.

Definition 5.11 A *consistent occurrence aliasing* is a partial injective function $g : \mathcal{O} \rightarrow \mathcal{O}$ which satisfies, for any e, e' on which it is defined:

(i) $\lambda(e) = \lambda(g(e))$

(ii) $e' \prec e \;\Leftrightarrow\; g(e') \prec g(e)$

Let $\mathcal{G}$ be the set of consistent occurrence aliasings on $\mathcal{O}$.

Definition 5.12 A *progressive o-bisimulation family* is a $\mathcal{G}$-indexed family of relations over $\mathcal{S}$, $\mathbf{R} = \{R_g \mid g \in \mathcal{G}\}$, such that if $\xi_0 \, R_g \, \xi'_0$ then for all $e \in \mathcal{O}$:

(1) $\xi_0 \overset{e}{\Longrightarrow} \xi_1 \;\Rightarrow\; \exists\, e', \xi'_1$ s.t. $\xi'_0 \overset{e'}{\Longrightarrow} \xi'_1$, $g \cup \{(e, e')\} \in \mathcal{G}$ and $\xi_1 \, R_{g \cup \{(e,e')\}} \, \xi'_1$

(2) $\xi'_0 \overset{e'}{\Longrightarrow} \xi'_1 \;\Rightarrow\; \exists\, e, \xi_1$ s.t. $\xi_0 \overset{e}{\Longrightarrow} \xi_1$, $g \cup \{(e, e')\} \in \mathcal{G}$ and $\xi_1 \, R_{g \cup \{(e,e')\}} \, \xi'_1$

(3) $\xi_0 \overset{\tau}{\Longrightarrow} \xi_1 \;\Rightarrow\; \exists \xi'_1 \in \mathcal{S}$ such that $\xi'_0 \overset{\tau}{\Longrightarrow} \xi'_1$ and $\xi_1 R_g \xi'_1$

(4) $\xi'_0 \overset{\tau}{\Longrightarrow} \xi'_1 \;\Rightarrow\; \exists \xi_1 \in \mathcal{S}$ such that $\xi_0 \overset{\tau}{\Longrightarrow} \xi_1$ and $\xi_1 R_g \xi'_1$

These relations induce an equivalence $\approx^{occ}$ on CCS processes as follows:

Definition 5.13 (Equivalence on the occurrence system) For any $p, q \in$ CCS, let $p \approx^{occ} q$ iff $pR_{\emptyset}q$ for some progressive o-bisimulation family $\mathbf{R} = \{R_g \mid g \in \mathcal{G}\}$.

The reader familiar with the notion of *history preserving bisimulation* (see e.g. [GG89]) may have noticed the similarity with our definition of $\approx^{occ}$. In fact history preserving bisimulation is itself a "progressive" notion, and it is clear that a consistent occurrence aliasing g is nothing else than an isomorphism between two partially ordered sets of occurrences. In the Appendix we shall give a definition of *local history preserving bisimulation* on the occurrence system (so-called because the ordering is that of local causality), and show that it is a direct reformulation of the equivalence $\approx^{occ}$.

A preorder $\sqsubseteq\!\!\!\!\raisebox{-1ex}{$\sim^{occ} is obtained by the same definition, after weakening the notion of *consistency* as follows:

Definition 5.14 A *right-consistent occurrence aliasing* is a partial injective function $g : \mathcal{O} \rightarrow \mathcal{O}$ which satisfies, for any e, e' on which it is defined:

(i) $\lambda(e) = \lambda(g(e))$

(ii) $g(e') \prec g(e) \;\Rightarrow\; e' \prec e$

Our main result is that the equivalence $\approx^{occ}$ coincides with both $\approx^s_\ell$ and $\approx^d_\ell$, and similarly that $\sqsubseteq\!\!\!\!\raisebox{-1ex}{$\sim^{occ} coincides with both $\sqsubseteq\!\!\!\!\raisebox{-1ex}{$\sim$}^{s}_{\ell}$ and $\sqsubseteq\!\!\!\!\raisebox{-1ex}{$\sim$}^{d}_{\ell}$. The proofs rely on the above properties of the occurrence system, and on proving conversion lemmas between the different kinds of transitions.

5.2 Occurrence semantics = static location semantics

We establish here the relationship between $\approx^{occ}$ and $\approx^s_\ell$. We saw in Section 3 that $\approx^s_\ell$ can be defined in terms of canonical distributions. We recall that these are distributions always associating location 0 to the left operand and 1 to the right operand of a parallel composition. Let CDIS denote the set of these canonical distributions, and $\eta, \zeta \in \{0,1\}^*$ range over canonical locations. With each state ξ, we associate a distributed process $dis(\xi) \in \mathrm{CDIS}$ as follows:

$$\begin{array}{lcl}
dis(\xi) & = & \xi, \ \text{ if } \xi = nil \ \text{ or } \ \xi = x \\
dis(a.p) & = & a.\,dis(p) \\
dis(p+q) & = & dis(p) + dis(q) \\
dis(rec\,x.\ p) & = & rec\,x.\ dis(p) \\
dis(\widehat{a}.\xi) & = & dis(\xi) \\
dis(\xi \mid \xi') & = & 0 :: dis(\xi) \mid 1 :: dis(\xi') \\
dis(\xi \backslash \alpha) & = & dis(\xi)\backslash \alpha \\
dis(\xi \langle f \rangle) & = & dis(\xi) \langle f \rangle
\end{array}$$

Thus $dis(\xi)$ is the canonical distribution of the CCS term underlying ξ. We can now give the conversion lemma between occurrence transitions and static location transitions:

Lemma 5.15 (**Conversion : static $\leftrightarrow$ occurrence**) *Let* $p, p' \in \mathrm{CDIS}$, $\xi, \xi' \in \mathcal{S}$. *Then:*

$$\begin{array}{llcl}
(i) & \xi \xrightarrow{\tau} \xi' & \Rightarrow & \exists \eta \ \text{ s.t. } \ dis(\xi) \ \underset{\eta}{\xrightarrow{\tau}}_s \ dis(\xi') \\
(ii) & p \ \underset{\eta}{\xrightarrow{\tau}}_s \ p' & \Rightarrow & \forall \xi \ \text{ s.t. } \ dis(\xi) = p \ \ \exists \xi' \ \text{ s.t. } \ \xi \xrightarrow{\tau} \xi' \ \text{ and } \ dis(\xi') = p' \\
(iii) & \xi \xrightarrow{e} \xi' & \Rightarrow & dis(\xi) \ \underset{\eta}{\xrightarrow{a}}_s \ dis(\xi'), \ \text{ where } \ a = \lambda(e) \ \text{ and } \ \eta = loc(e) \\
(iv) & p \ \underset{\eta}{\xrightarrow{a}}_s \ p' & \Rightarrow & \forall \xi \ \text{ s.t. } \ dis(\xi) = p \ \ \exists\, e, \exists \xi' \ \text{ s.t. } \ \lambda(e) = a, \ loc(e) = \eta, \\
 & & & \qquad dis(\xi') = p', \ \xi \xrightarrow{e} \xi'
\end{array}$$

In the proof of Lemma 5.15 we will use the following fact:

Fact 5.16 *Let* $p \in \mathrm{CCS}$. *Then:* $dis(p\,[rec\,x.\ p/x]) = dis(p)[rec\,x.\ dis(p)/x]$.

Proof: We show the more general statement:

$$\forall p,\, r \in \mathrm{CCS}: \ \ dis(p\,[rec\,x.\ r/x]) = dis(p)[rec\,x.\ dis(r)/x]$$

by induction on the structure of p. We only consider a couple of cases.

– *Basic cases:* $p = nil$, $p = x$. The first case is trivial. In the second case, we have immediately: $dis(x\,[rec\,x.\,r/x]) = dis(rec\,x.\,r) = rec\,x.\ dis(r) = x[rec\,x.\ dis(r)/x] = dis(x)[rec\,x.\ dis(r)/x]$.

– $p = rec\,x.\ q$ has no free occurrences of x. Thus $dis((rec\,x.\ q)[rec\,x.\ r/x]) = dis(rec\,x.\ q) = dis(rec\,x.\ q)[rec\,x.\ dis(r)/x]$.

– $p = rec\,y.\ q,\ \ y \neq x$. Then $dis((rec\,y.\ q)[rec\,x.\ r/x]) = dis(rec\,y.\ q[rec\,x.\ r/x]) = rec\,y.\ dis(q[rec\,x.\ r/x])$, which by induction equals $rec\,y.\ (dis(q))[rec\,x.\ dis(r)/x] = (rec\,y.\ dis(q))[rec\,x.\ dis(r)/x] = (dis(rec\,y.\ q))[rec\,x.\ dis(r)/x]$.

□

We give now the proof of the above Lemma.

Proof of Lemma 5.15: We start by proving (iii) and (iv), since they are needed to deal with the communication case in (i) and (ii). All clauses are proved by induction on the inference of the transition.

Proof of (iii). Consider the last rule applied to infer $\xi \xrightarrow{e} \xi'$.

– (O1) $a.p \xrightarrow{a} \widehat{a}.p$. We then have immediately: $dis(a.p) = a.\ dis(p) \xrightarrow[\varepsilon]{a}_s dis(p) = dis(\widehat{a}.p)$.

– (O2) $\widehat{a}.\xi \xrightarrow{a\,e} \widehat{a}.\xi'$ is inferred from $\xi \xrightarrow{e} \xi',\ e \neq \tau$. By induction $dis(\xi) \xrightarrow[loc(e)]{\lambda(e)}_s dis(\xi')$. Since $dis(\widehat{a}.\xi) = dis(\xi),\ dis(\widehat{a}.\xi') = dis(\xi')$ and $\lambda(a\,e) = \lambda(e),\ loc(a\,e) = loc(e)$, we have the required transition $dis(\widehat{a}.\xi) \xrightarrow[loc(a\,e)]{\lambda(a\,e)}_s dis(\widehat{a}.\xi')$.

– (O3) $\xi \mid \xi'' \xrightarrow{0\,e} \xi' \mid \xi''$ is inferred from $\xi \xrightarrow{e} \xi',\ e \neq \tau$. By induction we have $dis(\xi) \xrightarrow[loc(e)]{\lambda(e)}_s dis(\xi')$. Then $dis(\xi \mid \xi'') = 0 :: dis(\xi) \mid 1 :: dis(\xi'') \xrightarrow[0\,loc(e)]{\lambda(e)}_s 0 :: dis(\xi') \mid 1 :: dis(\xi'')$, which is the required transition since $\lambda(e) = \lambda(0\,e),\ 0\,loc(e) = loc(0\,e)$ and $0 :: dis(\xi') \mid 1 :: dis(\xi'') = dis(\xi' \mid \xi'')$.

– (O5), (O6), (O7): easy induction.

– (O8) $rec\,x.\ p \xrightarrow{e} \xi$ is inferred from $p\,[rec\,x.\ p/x] \xrightarrow{e} \xi$. By induction we have $dis(p\,[rec\,x.\ p/x]) \xrightarrow[loc(e)]{\lambda(e)}_s dis(\xi)$. By Fact 5.16 $dis(p\,[rec\,x.\ p/x]) = dis(p)[rec\,x.\ dis(p)/x])$. Then from $dis(p)[rec\,x.\ dis(p)/x]) \xrightarrow[loc(e)]{\lambda(e)}_s dis(\xi)$ we may deduce the required transition $dis(rec\,x.\ p) = rec\,x.\ dis(p) \xrightarrow[loc(e)]{\lambda(e)}_s dis(\xi)$.

Proof of (iv). Again, we consider the last rule applied in the proof of the transition $p \xrightarrow[\eta]{a}_s p'$.

– $a.p \xrightarrow[\varepsilon]{a}_s p$. Let $\xi \in \mathcal{S}$ be such that $dis(\xi) = a.p$. Since dis preserves all the operators except the "hats", which are erased, either $\xi = a.p$ or ξ is of the form $\widehat{a_1}.\ \ldots.\widehat{a_n}.a.p$, in short $\widehat{\sigma}.a.p$. In the first case we have $a.p \xrightarrow{a} \widehat{a}.p$, which clearly satisfies the conditions. In the second case, by iterated use of (O2) we derive $\widehat{\sigma}.\, a.\, p \xrightarrow{\sigma\, a} \widehat{\sigma}.\, \widehat{a}.\, p$, $\sigma \in Act^*$, which also satisfies the conditions.

– $0 :: p_0 \mid 1 :: p_1 \xrightarrow[0\,\eta]{a}_s 0 :: p_0' \mid 1 :: p_1$ is inferred from $p_0 \xrightarrow[\eta]{a}_s p_0'$. Note that any ξ such that $dis(\xi) = 0 :: p_0 \mid 1 :: p_1$ is necessarily of the form $\xi_0 \mid \xi_1$, where $dis(\xi_i) = p_i$. By induction $\exists\, e, \exists\, \xi_0'$ s.t. $\lambda(e) = a$, $loc(e) = \eta$, $dis(\xi_0') = p_0'$. Then we have a transition $\xi_0 \mid \xi_1 \xrightarrow{0\,e} \xi_0' \mid \xi_1$, which is the required one since $\lambda(0\,e) = \lambda(e) = a$, $loc(0\,e) = 0\, loc(e) = 0\,\eta$.

– $p_0 + p_1 \xrightarrow[\eta]{a}_s p_0'$ is inferred from $p_0 \xrightarrow[\eta]{a}_s p_0'$. Here ξ is of the form $q_0 + q_1$, where for each $i:\ q_i \in$ CCS and $dis(q_i) = p_i$. Actually $\xi = \pi(p_0) + \pi(p_1)$ – recall from Section 3 that for any distributed process p, $\pi(p)$ is the function yielding the underlying CCS process. Now by induction $\exists\, e, \exists\, \xi'$ s.t. $\lambda(e) = a$, $loc(e) = \eta$, $dis(\xi') = p_0'$ and $q_0 \xrightarrow{e} \xi'$. Then also $\xi = q_0 + q_1 \xrightarrow{e} \xi'$.

– $rec\, x.\ p \xrightarrow[\eta]{a}_s p'$ is inferred from $p\,[rec\, x.\ p/x] \xrightarrow[\eta]{a}_s p'$. Here ξ is of the form $rec\, x.\ q$, where $q \in$ CCS and $dis(q) = p$. Actually, $\xi = rec\, x.\ \pi(p)$. Now by Fact 5.16 we have $p\,[rec\, x.\ p/x] = dis(q)[rec\, x.\ dis(q)/x]) = dis(q\,[rec\, x.\ q/x])$. Let $\xi'' = q[rec\, x.\ q/x]$. Since $dis(\xi'') = p\,[rec\, x.\ p/x]$, we know by induction that $\exists\, e, \exists\, \xi'$ such that $q\,[rec\, x.\ q/x] \xrightarrow{e} \xi'$, where $\lambda(e) = a$, $loc(e) = \eta$, $dis(\xi') = p'$. Then also $\xi = rec\, x.\ q \xrightarrow{e} \xi'$.

Proof of (i). We only consider the communication case, as the other cases are straightforward. So suppose the last rule applied for deriving $\xi \xrightarrow{\tau} \xi'$ is (O4).

– (O4) $\xi = \xi_0 \mid \xi_1 \xrightarrow{\tau} \delta_{e_0}(\xi_0') \mid \delta_{e_1}(\xi_1')$ is deduced from $\xi_0 \xrightarrow{e_0} \xi_0'$, $\xi_1 \xrightarrow{e_1} \xi_1'$, $\lambda(e_0) = \overline{\lambda(e_1)}$. Then by point (iii) above $dis(\xi_i) \xrightarrow[loc(e_i)]{\lambda(e_i)}_s dis(\xi_i')$, and thus by rule (S4) $dis(\xi_0 \mid \xi_1) = 0 :: dis(\xi_0) \mid 1 :: dis(\xi_1) \xrightarrow[\varepsilon]{\tau}_s 0 :: dis(\xi_0') \mid 1 :: dis(\xi_1')$. Now, since $dis(\widehat{a}.\, \xi) = dis(\xi)$ it should be clear that $dis(\xi_i') = dis(\delta_{e_i}(\xi_i'))$. Thus $0 :: dis(\xi_0') \mid 1 :: dis(\xi_1') = 0 :: dis(\delta_{e_0}(\xi_0')) \mid 1 :: dis(\delta_{e_1}(\xi_1')) = dis(\delta_{e_0}(\xi_0') \mid \delta_{e_1}(\xi_1'))$.

Proof of (ii). Again, we just consider the communication case. Suppose (S4) is the last rule applied in the proof of $p \xrightarrow[\eta]{\tau}_s p'$.

– (S4) $0 :: p_0 \mid 1 :: p_1 \xrightarrow[\varepsilon]{\tau}_s 0 :: p_0' \mid 1 :: p_1'$ is deduced from $p_0 \xrightarrow[\eta_0]{\alpha_0}_s p_0'$, $p_1 \xrightarrow[\eta_1]{\alpha_1}_s p_1'$, $\alpha_0 = \overline{\alpha_1}$. In this case $\xi = \xi_0 \mid \xi_1$ for some ξ_0, ξ_1 such that $dis(\xi_i) = p_i$. By point (iv) for each $i = 0, 1 :\ \exists\, e_i, \exists\, \xi_i'$ such that $loc(e_i) = \eta_i$, $dis(\xi_i') = p_i'$, $\lambda(e_i) = \alpha_i$ and $\xi_i \xrightarrow{e_i} \xi_i'$. Then by rule (O4) we deduce $\xi_0 \mid \xi_1 \xrightarrow{\tau} \delta_{e_0}(\xi_0') \mid \delta_{e_1}(\xi_1')$, which is the required transition since $dis(\delta_{e_0}(\xi_0') \mid \delta_{e_1}(\xi_1')) = dis(\xi_0' \mid \xi_1')$, as just shown in the proof of (i). □

As a straightforward corollary we have an analogous conversion for the weak transitions.

Lemma 5.17 (**Weak conversion**) *Let* $p, p' \in \mathrm{CDIS}$ *and* $\xi, \xi' \in \mathcal{S}$. *Then:*

$$\begin{array}{llll}
(i) & \xi \overset{\tau}{\Longrightarrow} \xi' & \Rightarrow & dis(\xi) \overset{\tau}{\Longrightarrow}_s dis(\xi') \\
(ii) & p \overset{\tau}{\Longrightarrow}_s p' & \Rightarrow & \forall \xi \text{ s.t. } dis(\xi) = p \;\; \exists \xi' \text{ s.t. } \xi \overset{\tau}{\Longrightarrow} \xi' \text{ and } dis(\xi') = p' \\
(iii) & \xi \overset{e}{\Longrightarrow} \xi' & \Rightarrow & dis(\xi) \underset{\eta}{\overset{a}{\Longrightarrow}}_s dis(\xi'), \text{ where } a = \lambda(e) \text{ and } \eta = loc(e) \\
(iv) & p \underset{\eta}{\overset{a}{\Longrightarrow}}_s p' & \Rightarrow & \forall \xi \text{ s.t. } dis(\xi) = p \;\; \exists\, e, \xi' \text{ s.t. } \lambda(e) = a,\ loc(e) = \eta, \\
 & & & \qquad dis(\xi') = p' \text{ and } \xi \overset{e}{\Longrightarrow} \xi'
\end{array}$$

Using Lemma 5.17 and Proposition 5.10 we may now prove our equivalence result.

Theorem 5.18 *For any* $p, q \in \mathrm{CCS}$: $p \approx^s_\ell q \;\Rightarrow\; p \approx^{occ} q$.

Proof: Suppose $p \approx^s_\ell q$, and let $\mathbf{S} = \{S_\varphi \mid \varphi \in \Phi\}$ be a progressive bisimulation family such that $dis(p)\, S_\emptyset\, dis(q)$. Define a family of relations $\mathbf{R} = \{R_g \mid g \in \mathcal{G}\}$ by:

$$\begin{array}{lll}
(\xi, \xi') \in R_g & \Longleftrightarrow & \text{there exist occurrence transition sequences} \\
 & & \qquad p \overset{\tau}{\Longrightarrow} \xi_0 \overset{e_1}{\Longrightarrow} \cdots \overset{e_n}{\Longrightarrow} \xi_n = \xi \\
 & & \qquad q \overset{\tau}{\Longrightarrow} \xi'_0 \overset{e'_1}{\Longrightarrow} \cdots \overset{e'_n}{\Longrightarrow} \xi'_n = \xi' \\
 & & \text{such that} \\
 & & (1)\;\; g \;=\; \{ (e_1, e'_1), \ldots, (e_n, e'_n) \} \\
 & & (2)\;\; dis(\xi)\; S_\varphi\; dis(\xi'),\;\; \text{where } \varphi = \{ (loc(e), loc(e')) \mid (e, e') \in g \}
\end{array}$$

Clearly $(p, q) \in R_\emptyset$. We show that $\mathbf{R}$ is a progressive o-bisimulation family. Assume $(\xi, \xi') \in R_g$.

- Let $\xi \overset{e}{\Longrightarrow} \xi_{n+1}$, $e \neq \tau$. By Lemma 5.17 (iii) $dis(\xi) \underset{\eta}{\overset{a}{\Longrightarrow}}_s dis(\xi_{n+1})$, where $a = \lambda(e)$ and $\eta = loc(e)$. Let $r = dis(\xi)$, $r_{n+1} = dis(\xi_{n+1})$ and $s = dis(\xi')$. Since $r\, S_\varphi\, s$, there exist η', s_{n+1} such that $s \underset{\eta'}{\overset{a}{\Longrightarrow}}_s s_{n+1}$ and $r_{n+1}\; S_{\varphi \cup (\eta, \eta')}\; s_{n+1}$, where $\varphi \cup (\eta, \eta')$ is a consistent location association. By Lemma 5.17 (iv), there exist now e', ξ'_{n+1} such that $\lambda(e') = a$, $loc(e') = \eta'$, $dis(\xi'_{n+1}) = s_{n+1}$ and $\xi' \overset{e'}{\Longrightarrow} \xi'_{n+1}$. We want to show that $g' = g \cup (e, e')$ is a consistent occurrence aliasing. Since g is consistent, and by Corollary 5.9(2) we have $\forall i : e \not\prec e_i, e' \not\prec e'_i$, we only have to check that $\forall i : e_i \prec e \Leftrightarrow e'_i \prec e'$. Since $\varphi \cup (\eta, \eta')$ is a consistent location association, we have $\forall i$: $loc(e_i) \diamond loc(e) \Leftrightarrow loc(e'_i) \diamond loc(e')$. This is equivalent to $loc(e_i) \ll loc(e) \Leftrightarrow loc(e'_i) \ll loc(e')$ by Corollary 5.9(2). Using Proposition 5.10, we may then conclude that $\forall i : e_i \prec e \Leftrightarrow e'_i \prec e'$.

- Let $\xi \overset{\tau}{\Longrightarrow} \xi_{n+1}$. By Lemma 5.17 (i) $dis(\xi) \overset{\tau}{\Longrightarrow}_s dis(\xi_{n+1})$. Let $r = dis(\xi)$, $r_{n+1} = dis(\xi_{n+1})$ and $s = dis(\xi')$. Since $r\, S_\varphi\, s$, there exists s_{n+1} such that $s \overset{\tau}{\Longrightarrow}_s s_{n+1}$ and $r_{n+1}\, S_\varphi\, s_{n+1}$. By Lemma 5.17 (ii) there exists ξ'_{n+1} such that $dis(\xi'_{n+1}) = s_{n+1}$ and

$\xi' \overset{\tau}{\Longrightarrow} \xi'_{n+1}$. Since the computations $p \overset{\tau}{\Longrightarrow} \xi_0 \overset{e_1}{\Longrightarrow} \cdots \overset{e_n}{\Longrightarrow} \xi_{n+1}$ and $q \overset{\tau}{\Longrightarrow} \xi'_0 \overset{e'_1}{\Longrightarrow} \cdots \overset{e'_n}{\Longrightarrow} \xi'_{n+1}$ are of the required form, we have then $(\xi_{n+1}, \xi'_{n+1}) \in R_g$. The initial $\overset{\tau}{\Longrightarrow}$ in the computations is needed to cover the case $\xi_0 = \xi \overset{\tau}{\Longrightarrow} \xi_1$. □

Theorem 5.19 *For any* $p, q \in \mathrm{CCS}$*:* $\quad p \approx^{occ} q \;\;\Rightarrow\;\; p \approx^s_\ell q$.

Proof: Suppose $p \approx^{occ} q$, and let $\mathbf{R} = \{R_g \mid g \in \mathcal{G}\}$ be a progressive o-bisimulation family such that $p\, R_\emptyset\, q$. Consider the family $\mathbf{S} = \{S_\varphi \mid \varphi \in \Phi\}$ of relations over CDIS given by:

$$(r,s) \in S_\varphi \iff \text{there exist occurrence transition sequences}$$

$$p \overset{\tau}{\Longrightarrow} \xi_0 \overset{e_1}{\Longrightarrow} \cdots \overset{e_n}{\Longrightarrow} \xi_n = \xi$$

$$q \overset{\tau}{\Longrightarrow} \xi'_0 \overset{e'_1}{\Longrightarrow} \cdots \overset{e'_n}{\Longrightarrow} \xi'_n = \xi'$$

such that

(1) $dis(\xi) = r\,, \quad dis(\xi') = s$

(2) $\varphi = \{\, (\, loc(e_i), loc(e'_i)\,) \mid 1 \le i \le n \,\}$

(3) $\xi\, R_g\, \xi'$ for $g = \{\, (e_i, e'_i) \mid 1 \le i \le n \,\}$

We show that $\mathbf{S}$ is a progressive bisimulation family. Assume $(r,s) \in S_\varphi$.

- Let $r \underset{\eta}{\overset{a}{\Longrightarrow}}_s r_{n+1}$. By Lemma 5.17 (iv), there exist e, ξ_{n+1} such that $\lambda(e) = a$, $loc(e) = \eta$, $dis(\xi_{n+1}) = r_{n+1}$ and $\xi \overset{e}{\Longrightarrow} \xi_{n+1}$. Since $\xi\, R_g\, \xi'$ there exist now e', ξ'_{n+1} such that $g \cup (e,e')$ is a consistent occurrence aliasing and $\xi' \overset{e'}{\Longrightarrow} \xi'_{n+1}$. The consistency of $g \cup (e,e')$ implies $\lambda(e') = \lambda(e) = a$. By Lemma 5.17 (iii) we have then $dis(\xi') \underset{\eta'}{\overset{a}{\Longrightarrow}}_s dis(\xi'_{n+1})$, where $\eta' = loc(e')$. We want to show that $\varphi \cup (\eta, \eta')$ is a consistent location association. Since φ is supposed to be consistent, we only have to check that for any i: $loc(e_i) \diamond loc(e) \;\Leftrightarrow\; loc(e'_i) \diamond loc(e')$. Since $g \cup (e,e')$ is consistent we know that $e_i \prec e \;\Leftrightarrow\; e'_i \prec e'$. But since e_i, e'_i occur respectively before e, e', this is equivalent, by Proposition 5.10, to $loc(e_i) \ll loc(e) \;\Leftrightarrow\; loc(e'_i) \ll loc(e')$, which in turn is equivalent, by Corollary 5.9, to $loc(e_i) \diamond loc(e) \;\Leftrightarrow\; loc(e'_i) \diamond loc(e')$.

- The case $r \overset{\tau}{\Longrightarrow}_s r_{n+1}$ is straightforward, applying Lemma 5.17 (ii) and Lemma 5.17 (i). □

We prove now the coincidence of the preorders:

Theorem 5.20 *For any* $p, q \in \mathrm{CCS}$*:* $p \sqsubseteq\!\!\!\!\sim_{\ell}^{s} q \;\Leftrightarrow\; p \sqsubseteq\!\!\!\!\sim^{occ} q$.

PROOF: $\Rightarrow$: Variation of the proof of Theorem 5.18. We take a progressive prebisimulation family $\mathbf{S}$, and show that $\mathbf{R}$, as defined above, is a progressive o-prebisimulation family. To show that $g' = g \cup (e, e')$ is a right-consistent occurrence aliasing, we need to check that $\forall i : e'_i \prec e' \;\Rightarrow\; e_i \prec e$. Since $\varphi \cup (\eta, \eta')$ is a left-consistent location association, we know that $\forall i$: $loc(e_i) \diamond loc(e) \;\Rightarrow\; loc(e'_i) \diamond loc(e')$. By Corollary 5.9(2), this is equivalent to $loc(e'_i) \ll loc(e') \;\Rightarrow\; loc(e_i) \ll loc(e)$. By Proposition 5.10, we have then $\forall i : e'_i \prec e' \;\Rightarrow\; e_i \prec e$.

$\Leftarrow$: Adaptation of the proof of Theorem 5.19. To show that $\varphi \cup (\eta, \eta')$ is a left-consistent location association, we check that $\forall i$: $loc(e_i) \diamond loc(e) \;\Rightarrow\; loc(e'_i) \diamond loc(e')$. Since $g \cup (e, e')$ is right-consistent we know that $e'_i \prec e' \;\Rightarrow\; e_i \prec e$. This is equivalent to $loc(e'_i) \ll loc(e') \;\Rightarrow\; loc(e_i) \ll loc(e)$, which in turn is equivalent to $loc(e_i) \diamond loc(e) \;\Rightarrow\; loc(e'_i) \diamond loc(e')$. $\square$

5.3 Occurrence semantics = dynamic location semantics

We turn now to the relation between $\approx^{occ}$ and $\approx_{\ell}^{d}$. To establish the coincidence of the two equivalences, we will use the fact that $\approx_{\ell}^{d}$ may be obtained by restricting attention to computations where distinct atomic locations are chosen at each step. This fact was first pointed out by Kiehn in [Kie91]. Let us recall some definitions and results from [BCHK91]:

Definition 5.21 A *location renaming* is a mapping $\rho : Loc \rightarrow Loc^*$. For any $p \in$ LCCS, let $p[\rho]$ denote the process obtained by replacing all occurrences of l in p with $\rho(l)$, for any $l \in Loc$.

We use the notation $\rho\{u/l\}$ for the renaming which maps l to u and acts like ρ on $Loc \setminus \{l\}$. Also, we shall abbreviate $p[id\{u/l\}]$ to $p\{u/l\}$. In what follows, we shall mainly consider alphabetic renamings $\rho : Loc \rightarrow Loc$.

Note that any partial function $f : Loc \rightarrow Loc$ may be seen as a location renaming $\rho : Loc \rightarrow Loc$, by letting:

$$\rho(l) \;=\; \begin{cases} l & \text{if } f(l) \text{ is not defined} \\ f(l) & \text{otherwise.} \end{cases}$$

For instance the empty function $\emptyset$ corresponds to the identity renaming id. In the following we shall freely use the renaming notation $p[f]$ whenever f is a partial function $f : Loc \rightarrow Loc$.

The following lemma (similar to those of [BCHK93], [BCHK91]), relates the transitions of $p[\rho]$ with those of p.

Lemma 5.22 *Let $p \in LCCS$, and $\rho : Loc \to Loc$ be an alphabetic location renaming. Then:*

1. a) $p \xrightarrow[u]{\tau}_d p' \;\Rightarrow\; \exists v$ *s.t.* $\rho(u) \ll v$ *and* $p[\rho] \xrightarrow[v]{\tau}_d p'[\rho]$.

   b) $p \overset{\tau}{\Longrightarrow}_d p' \;\Rightarrow\; p[\rho] \overset{\tau}{\Longrightarrow}_d p'[\rho]$.

2. a) $p[\rho] \xrightarrow[v]{\tau}_d p' \;\Rightarrow\; \exists u, p''$ *such that* $\rho(u) \ll v$, $p''[\rho] = p'$ *and* $p \xrightarrow[v]{\tau}_d p''$.

   b) $p[\rho] \overset{\tau}{\Longrightarrow}_d p' \;\Rightarrow\; \exists p''$ *such that* $p''[\rho] = p'$ *and* $p \overset{\tau}{\Longrightarrow}_d p''$.

3. a) $p \xrightarrow[ul]{a}_d p'$, $l \notin Loc(p) \;\Rightarrow\; \forall k \in Loc$, $p[\rho] \xrightarrow[\rho(u)\,k]{a}_d p'[\rho\{k/l\}]$.

   b) *Same as a), with weak transitions.*

4. a) $p[\rho] \xrightarrow[vl]{a}_d p' \;\Rightarrow\; \exists\, u$ *such that* $\rho(u) = v$ *and* $\forall\, k \notin Loc(p)\; \exists\, p''$ *such that* $p''[\rho\{l/k\}] = p'$ *and* $p \xrightarrow[uk]{a}_d p''$.

   b) *Same as a), with weak transitions.*

We recall now Kiehn's definition for $\approx_\ell^d$, and show that it is equivalent to the original one.

Notation 5.23 *Let $LCCS^\nu$ be the set of LCCS processes whose atomic locations are all distinct.*

Definition 5.24 (ν–**dynamic location equivalence** [Kie91])

A relation $R \subseteq (\mathrm{LCCS}^\nu \times \mathrm{LCCS}^\nu)$ is a *ν–dynamic location bisimulation (ν–dlb)* iff for all $(p, q) \in R$ and for all $a \in Act, u \in Loc^*$:

(1) $p \overset{a}{\underset{ul}{\Longrightarrow}}_d p'$, $l \notin Loc(p) \cup Loc(q) \;\Rightarrow\; \exists\, q'$ s.t. $q \overset{a}{\underset{ul}{\Longrightarrow}}_d q'$ and $(p', q') \in R$

(2) $q \overset{a}{\underset{ul}{\Longrightarrow}}_d q'$, $l \notin Loc(p) \cup Loc(q) \;\Rightarrow\; \exists\, p'$ s.t. $p \overset{a}{\underset{ul}{\Longrightarrow}}_d p'$ and $(p', q') \in R$

(3) $p \overset{\tau}{\Longrightarrow}_d p' \;\Rightarrow\; \exists\, q'$ s.t. $q \overset{\tau}{\Longrightarrow}_d q'$ and $(p', q') \in R$

(4) $q \overset{\tau}{\Longrightarrow}_d q' \;\Rightarrow\; \exists\, p'$ s.t. that $p \overset{\tau}{\Longrightarrow}_d p'$ and $(p', q') \in R$

The largest ν–*dlb* is called *ν–dynamic location equivalence* and denoted $\approx_\ell^\nu$.

Fact 5.25 *For any processes $p, q \in CCS$: $p \approx_\ell^d q \;\Leftrightarrow\; p \approx_\ell^\nu q$.*

Proof: $\Rightarrow$: trivial. $\Leftarrow$: Let $p \approx_\ell^\nu q$. Then there exists a ν–*dlb* R s.t. $p\,R\,q$. Define now:

$$S = \{\, (r[\rho], s[\rho]\,) \mid\; r \approx_\ell^\nu s,\; \rho\; Loc \to Loc\,\}$$

Clearly $p\,S\,q$, for $\rho = id$. We show that S is a *dlb*. Suppose $r[\rho]\;S\;s[\rho]$.

– Let $r[\rho] \overset{a}{\underset{ul}{\Longrightarrow}}_d r'$, l not necessarily new. Take $k \notin loc(r) \cup loc(s)$. By Lemma 5.22.(*4*), $\exists\, v,\ r''$ s.t. $r \overset{a}{\underset{vk}{\Longrightarrow}}_d r''$, $\rho(v) = u$ and $r''[\rho\{l/k\}] = r'$. Then, since $r \approx_\ell^\nu s$, there exists s' s.t. $s \overset{a}{\underset{vk}{\Longrightarrow}}_d s'$ and $r'' \approx_\ell^\nu s'$. By Lemma 5.22.(*2*), $s[\rho] \overset{a}{\underset{ul}{\Longrightarrow}}_d s'[\rho\{l/k\}]$. Since $r'' \approx_\ell^\nu s'$, we have $r''[\rho\{l/k\}]\ \ S\ \ s'[\rho\{l/k\}]$, hence S is a *dlb*. □

We proceed now to show that $\approx^{occ} = \approx_\ell^\nu$. To do this, we need to establish a conversion between occurrence transitions and dynamic location transitions. We start by converting terms ξ into LCCS terms which represent the same state of computation. The idea is to replace every "hat" in ξ by a canonical atomic location representing uniquely the corresponding occurrence. The simplest way to do this is to take the occurrences themselves as canonical locations. We shall then assume, from now onwards, that $\mathcal{O} \subseteq Loc$. We also introduce, for any $\gamma \in Act \cup \{0,1\}$, a renaming $\rho_\gamma(p)$ which prefixes by γ all the occurrences appearing as locations in p, namely $\rho_\gamma(e) = \gamma e$. Then the canonical LCCS process $proc(\xi)$ corresponding to a computation state $\xi \in \mathcal{S}$ is defined by:

$$\begin{array}{lcl} proc(\xi) & = & \xi, \text{ if } \xi \in \text{CCS} \\ proc(\widehat{a}.\,\xi) & = & a :: proc(\xi)[\rho_a] \\ proc(\xi \mid \xi') & = & proc(\xi)[\rho_0] \mid proc(\xi')[\rho_1] \\ proc(\xi \backslash \alpha) & = & proc(\xi)\backslash\alpha \\ proc(\xi\,\langle f \rangle) & = & proc(\xi)\,\langle f \rangle \end{array}$$

We have for instance: $proc(\widehat{a}.\,\widehat{b}.\,nil \mid c.\,nil) = 0a :: 0ab :: nil \mid c.\,nil$. It can be easily checked that $Loc(proc(\xi)) = occ\,(\xi)$ and $proc(\xi) \in \text{LCCS}^\nu$.

We introduce next some notation that will be used for proving the conversion lemma. The reader not interested in the details of the proof should proceed directly to the weak version of the conversion lemma (Lemma 5.28 at p. 36).

We define now, for $p \in \text{LCCS}^\nu$, a partial function $where\,(k,p)$ which gives the place where the atomic dynamic location k occurs in p, if it exists. This place is expressed as a static canonical location $\eta \in \{0,1\}^*$. The function $where\,(k,p)$ is essentially the same as that used by D. Yankelevich in [Yan93], p. 124. Here we only define it on LCCS^ν processes which can be reached from CCS processes (that is, where locations do not appear under recursion or the dynamic operators). The partial function $where : (Loc \times \text{LCCS}^\nu) \to \{0,1\}^*$ is given by:

$$where\,(k, l :: p) = \begin{cases} \varepsilon & \text{if } k = l \\ where\,(k,p) & \text{otherwise} \end{cases}$$

$$where\,(k, p \mid q) = \begin{cases} 0 \cdot where\,(k,p) & \text{if } where\,(k,p) \text{ is defined} \\ 1 \cdot where\,(k,q) & \text{if } where\,(k,q) \text{ is defined} \end{cases}$$

$$where\,(k, p\backslash\alpha) = where\,(k, p\,\langle f \rangle) = where\,(k,p)$$

In the coming lemma, we shall also use the function $\Delta_k(p)$ introduced in section 4. Recall that $\Delta_k(p)$ is the partial function that erases the location k in p, if it exists. We noted already the similarity between $\Delta_k(p)$ and the function $\delta_e(\xi)$ defined at p. 21. In fact, one has the following:

Remark 5.26 *For any* $\xi \in \mathcal{S}: \ proc(\delta_e(p)) = \Delta_e(proc(\xi))$.

The conversion lemma between occurrence transitions and dynamic location transitions is now:

Lemma 5.27 (**Conversion : dynamic $\leftrightarrow$ occurrence**)
Let $\xi, \xi' \in \mathcal{S}, \ f : occ(\xi) \rightarrow Loc$. *Then:*

$$
\begin{array}{llcl}
(i) & \xi \xrightarrow{\tau} \xi' & \Rightarrow & \exists\, u \ \text{ s.t. } \ proc(\xi)[f] \xrightarrow[u]{\tau}_d proc(\xi')[f] \\
(ii) & proc(\xi)[f] \xrightarrow[u]{\tau}_d p' & \Rightarrow & \exists\, \xi' \ \text{ s.t. } \ proc(\xi')[f] = p' \ \text{ and } \ \xi \xrightarrow{\tau} \xi' \\
(iii) & \xi \xrightarrow{e} \xi' & \Rightarrow & \forall\, l \in Loc: \ proc(\xi)[f] \xrightarrow[ul]{a}_d proc(\xi')\,[f\{l/e\}]\,, \\
 & & & \quad \text{where } \ a = \lambda(e)\,, \ u = f(path(e)) \\
(iv) & proc(\xi)[f] \xrightarrow[ul]{a}_d p' & \Rightarrow & \exists\, e, \xi' \ \text{ such that } \ \lambda(e) = a, \ f(path(e)) = u, \\
 & & & \quad proc(\xi')\,[f\{l/e\}] = p' \ \text{ and } \ \xi \xrightarrow{e} \xi'
\end{array}
$$

Proof: By induction on the proof of the transition in the hypothesis.

<u>*Proof of (iii)*</u>. Consider the last rule used to infer $\xi \xrightarrow{e} \xi'$. We take some representative cases.

– (O1) $a.p \xrightarrow{a} \widehat{a}.p$. Since $Loc(proc(a.p)) = \emptyset$, we have $proc(a.p)[f] = a.p$. By (D1) $a.p \xrightarrow[l]{a}_d l :: p$, for any $l \in Loc$. This is the required transition since $a = \lambda(a)$, $\varepsilon = f(\varepsilon) = f(path(a))$ and $proc(\widehat{a}.p)[f\{l/a\}] = (a :: (proc(p)[\rho_a]))[f\{l/a\}] = (a :: p)[f\{l/a\}] = l :: p$.

– (O2) $\widehat{a}.\xi \xrightarrow{a\,e} \widehat{a}.\xi'$ is inferred from $\xi \xrightarrow{e} \xi'$, $e \neq \tau$. By induction we have $proc(\xi) \xrightarrow[path(e)\cdot e]{\lambda(e)}_d proc(\xi')$, taking $l = e$ and $f = id \restriction occ(\xi)$. Note that $e \notin Loc(proc(\xi))$. Then, applying Lemma 5.22 (*2*) with $\rho = id$ and $k = ae$, and subsequently rule (S2), we derive a transition $proc(\widehat{a}.\xi) = a :: (proc(\xi)[\rho_a]) \xrightarrow[a\cdot\rho_a(path(e))\cdot ae]{\lambda(e)}_d a :: (proc(\xi')[\rho_a\{ae/e\}]) = a :: (proc(\xi')[\rho_a]) = proc(\widehat{a}.\xi')$. Note that $a \cdot \rho_a(path(e)) = path(ae)$. Now, from $e \notin Loc(proc(\xi))$ we deduce $ae \notin Loc(proc(\widehat{a}.\xi))$. Thus by Lemma 5.22 (*2*) again we obtain, for any $l \in Loc$, $proc(\widehat{a}.\xi)[f] \xrightarrow[f(path(ae))\cdot l]{\lambda(e)}_d proc(\widehat{a}.\xi')[f\{l/ae\}]$, which is the transition we sought for, since $\lambda(e) = \lambda(ae)$.

– (O8) $rec\,x.\ p \xrightarrow{e} \xi'$ is deduced from $p\,[rec\,x.\ p/x] \xrightarrow{e} \xi'$. By induction we have $proc(p\,[rec\,x.\ p/x]) \xrightarrow[path(e)\cdot e]{\lambda(e)}_d proc(\xi')$, taking $f = id$, $l =$

e. Since $proc(p\,[rec\,x.\ p/x]) = p\,[rec\,x.\ p/x]$ we have also, by rule (S8), a transition $proc(rec\,x.\ p) = rec\,x.\ p \xrightarrow[path(e)\cdot e]{\lambda(e)}_d proc(\xi')$. Whence, since $e \notin Loc(rec\,x.\ p)$, applying Lemma 5.22 (*2*) we obtain, for any $l \in Loc$, $proc(rec\,x.\ p)[f] \xrightarrow[f(path(e))\cdot l]{\lambda(e)}_d proc(\xi')[f\{l/e\}]$.

Proof of (*iv*). Let $proc(\xi)[f] \xrightarrow[ul]{a}_d p'$. By the renaming Lemma 5.22 (*4*) $\exists\, v$ s.t. $f(v) = u$ and $\forall k \notin loc(proc(\xi))\ \exists\, p_k$ s.t. $proc(\xi) \xrightarrow[vk]{a}_d p_k$ and $p_k[f\{l/k\}] = p'$. We show now, by induction on the proof of $proc(\xi) \xrightarrow[vk]{a}_d p_k$, that there exist $e,\ \xi'$ such that $\xi \xrightarrow{e} \xi'$, $\lambda(e) = a$, $path(e) = v$, $loc(e) = where\,(k, p_k)$ and $proc(\xi') = p_k\,\{e/k\}$. This will imply $proc(\xi')\,[f\{l/e\}] = p'$. The other conditions will also be satisfied, since $\lambda(e) = a$ and $f(path(e)) = f(v) = u$. We examine two representative cases.

– (D1) $proc(a.p) = a.p \xrightarrow[k]{a}_d k :: p = p_k$. By (O1) $a.p \xrightarrow{a} \widehat{a}.p$, where $\lambda(a) = a$, $path(a) = \varepsilon$, $loc(a) = \varepsilon = where\,(k, p_k)$ and $proc(\widehat{a}.p) = a :: p = p_k\,\{a/k\}$.

– (S2) $proc(\widehat{b}.\xi) = b :: (proc(\xi)[\rho_b]) \xrightarrow[b\cdot\rho_b(v)\cdot k]{a}_d b :: p' = p_k$ is deduced from $proc(\xi) \xrightarrow[v\cdot k']{a}_d q_{k'}$, where $k' \notin Loc(proc(\xi))$, $q_{k'}[\rho_b\{k/k'\}] = p'$. By induction $\exists\, e,\ \xi'$ such that $\xi \xrightarrow{e} \xi'$, $\lambda(e) = a$, $path(e) = v$, $loc(e) = where\,(k', q_{k'})$ and $proc(\xi') = q_{k'}\,\{e/k'\}$. By (O2) we have $\widehat{b}.\xi \xrightarrow{b\,e} \widehat{b}.\xi'$, which is the required transition since $\lambda(be) = \lambda(e) = a$, $path(be) = b \cdot \rho_b(path(e)) = b \cdot \rho_b(v)$, $loc(be) = loc(e) = where\,(k', q_{k'}) = where\,(k, q_{k'}[\rho_b\{k/k'\}]) = where\,(k, p') = where\,(k, p_k)$ and $proc(\widehat{b}.\xi') = b :: (proc(\xi')[\rho_b]) = b :: (q_{k'}\,\{e/k'\}[\rho_b]) = b :: (q_{k'}\,[\rho_b\{be/k'\}]) = b :: (q_{k'}\,[\rho_b\{k/k'\}]\{be/k\}) = p_k\{be/k\}$.

Note that the pair (e, ξ') thus determined is unique since, by Fact 5.5, an occurrence e is completely characterized by its label $\lambda(e)$, access path $path(e)$ and static location $loc(e)$.

Proof of (*i*). By induction on the proof of $\xi \xrightarrow{\tau} \xi'$. We only consider the communication case.

– (O4) $\xi_0 \mid \xi_1 \xrightarrow{\tau} \delta_{e_0}(\xi_0') \mid \delta_{e_1}(\xi_1')$ is deduced from $\xi_0 \xrightarrow{e_0} \xi_0'$, $\xi_1 \xrightarrow{e_1} \xi_1'$, $\lambda(e_0) = \overline{\lambda(e_1)}$. From $e_i \notin occ\,(\xi_i) = Loc(proc(\xi_i))$ we deduce $ie_i \notin Loc(proc(\xi_i)[\rho_i])$. Then by point (*iii*) above $proc(\xi_i)[\rho_i] \xrightarrow[\rho_i(path(e_i))\cdot i\, e_i]{\lambda(e_i)}_d proc(\xi_i')[\rho_i]$. Corresponding to these dynamic transitions we have, by Fact 4.1, the static transitions: $proc(\xi_i)[\rho_i] \xrightarrow[\rho_i(path(e_i))]{\lambda(e_i)}_s \Delta_{ie_i}(proc(\xi_i')[\rho_i]) = (\Delta_{e_i}(proc(\xi_i'))[\rho_i] = proc(\delta_{e_i}(\xi_i'))[\rho_i]$. From these, by rule (S4), we deduce $proc(\xi_0 \mid \xi_1) = proc(\xi_0)[\rho_0] \mid proc(\xi_1)[\rho_1] \xrightarrow[\varepsilon]{\tau}_d proc(\delta_{e_0}(\xi_0'))[\rho_0] \mid proc(\delta_{e_1}(\xi_1'))[\rho_1] = proc(\,\delta_{e_0}(\xi_0') \mid \delta_{e_1}(\xi_1')\,)$. Whence the result, by the renaming Lemma 5.22 (*2*).

Proof of (ii). Let $proc(\xi)[f] \xrightarrow[u]{\tau}_d p'$. By the renaming Lemma 5.22 (2) $\exists\, p''$ s.t. $proc(\xi) \xrightarrow[u]{\tau}_d p''$ and $p''[f] = p'$. We show, by induction on the proof of $proc(\xi) \xrightarrow[u]{\tau}_d p''$, that $\exists\, \xi''$ such that $\xi \xrightarrow{\tau} \xi''$ and $proc(\xi'') = p''$. This will imply $proc(\xi'')[f] = p''[f] = p'$. Again, we only consider communication.

– (S4) $proc(\xi_0 \mid \xi_1) = (\, proc(\xi_0)[\rho_0] \mid proc(\xi_1)[\rho_1]\,) \xrightarrow[\varepsilon]{\tau}_d q'_0 \mid q'_1$ is deduced from $proc(\xi_i)[\rho_i] \xrightarrow[u_i]{a_i}_s q'_i$, $u_0 \sqcap u_1 = u$, $a_0 = \overline{a_1}$. By Fact 4.1, for any $l_i \notin Loc(proc(\xi_i)[\rho_i])$ there exists q''_i such that $proc(\xi_i)[\rho_i] \xrightarrow[u_i \cdot l_i]{a_i}_d q''_i$ and $\Delta_{l_i}(q''_i) = q'_i$. By point (iv) there exist e_i, ξ'_i s.t. $\xi_i \xrightarrow{e_i} \xi'_i$, $\lambda(e_i) = a_i$, $path(e_i) = u_i$, $loc(e_i) = where(l_i, q''_i)$ and $proc(\xi'_i)[\rho_i\{l_i/e_i\}] = q''_i$. Then by rule (O4) $\xi_0 \mid \xi_1 \xrightarrow{\tau} \delta_{e_0}(\xi'_0) \mid \delta_{e_1}(\xi'_1)$. This is as required since $proc(\delta_{e_0}(\xi'_0) \mid \delta_{e_1}(\xi'_1)) = proc(\delta_{e_0}(\xi'_0))[\rho_0] \mid proc(\delta_{e_1}(\xi'_1))[\rho_1] = \Delta_{e_0}(proc(\xi'_0))[\rho_0] \mid \Delta_{e_1}(proc(\xi'_1))[\rho_1] = \Delta_{0e_0}(proc(\xi'_0)[\rho_0]) \mid \Delta_{1e_1}(proc(\xi'_1)[\rho_1])$, and the last term is equal to $\Delta_{l_0}(proc(\xi'_0)[\rho_0\{l_0/e_0\}]) \mid \Delta_{l_1}(proc(\xi'_1)[\rho_1\{l_1/e_1\}]) = q'_0 \mid q'_1$. □

We have now, as an immediate corollary:

Lemma 5.28 (**Weak conversion**) *Let $\xi, \xi' \in \mathcal{S}$, $f : occ(\xi) \to Loc$. Then:*

$$\begin{array}{llcl}
(i) & \xi \overset{\tau}{\Longrightarrow} \xi' & \Rightarrow & proc(\xi)[f] \overset{\tau}{\Longrightarrow}_d proc(\xi')[f] \\
(ii) & proc(\xi)[f] \overset{\tau}{\Longrightarrow}_d p' & \Rightarrow & \exists\, \xi' \text{ s.t. } proc(\xi')[f] = p' \text{ and } \xi \overset{\tau}{\Longrightarrow} \xi' \\
(iii) & \xi \overset{e}{\Longrightarrow} \xi' & \Rightarrow & \forall l \in Loc: \ proc(\xi)[f] \underset{ul}{\overset{a}{\Longrightarrow}}_d proc(\xi')\,[f\{l/e\}], \\
 & & & \text{where } \ a = \lambda(e),\ u = f(path(e)) \\
(iv) & proc(\xi)[f] \underset{ul}{\overset{a}{\Longrightarrow}}_d p' & \Rightarrow & \exists\, e, \xi' \text{ such that } \ \lambda(e) = a,\ f(path(e)) = u, \\
 & & & proc(\xi')\,[f\{l/e\}] = p' \text{ and } \ \xi \overset{e}{\Longrightarrow} \xi'
\end{array}$$

□

We have now all the elements to prove our results.

Theorem 5.29 *For any $p, q \in$ CCS: $\ p \approx^{occ} q \ \Rightarrow\ p \approx^{d}_{\ell} q$.*

Proof: Let $p \approx^{occ} q$. Then there exists a family of relations $\mathbf{R} = \{R_g \mid g \in \mathcal{G}\}$ such that $pR_\emptyset q$. Define a relation S on processes by:

$(r,s) \in S \iff$ there exist occurrence transition sequences

$$p \overset{\tau}{\Longrightarrow} \xi_0 \overset{e_1}{\Longrightarrow} \cdots \overset{e_n}{\Longrightarrow} \xi_n = \xi$$

$$q \overset{\tau}{\Longrightarrow} \xi'_0 \overset{e'_1}{\Longrightarrow} \cdots \overset{e'_n}{\Longrightarrow} \xi'_n = \xi'$$

and there exist two functions:

$$f_1 : \{e_1, \ldots, e_n\} \to Loc\,, \quad f_2 : \{e'_1, \ldots, e'_n\} \to Loc$$

such that

$$(1)\ \forall i \in \{1, \ldots, n\}:\ f_1(e_i) = f_2(e'_i)$$

$$(2)\ proc(\xi)\,[f_1] = r\,,\quad proc(\xi')[f_2] = s$$

$$(3)\ \xi\, R_g\, \xi' \ \text{ for } \ g = \{\, (e_1, e'_1), \ldots, (e_n, e'_n)\,\}$$

We show that S is a dynamic location bisimulation. Let $(r,s) \in S$.

- Suppose $r \underset{ul}{\overset{a}{\Longrightarrow}}_d r'$. By Lemma 5.28 (iv) there exist e, ξ_{n+1} such that $\lambda(e) = a$, $f_1(path(e)) = u$, $proc(\xi_{n+1})\,[f_1\{l/e\}] = r'$ and $\xi \overset{e}{\Longrightarrow} \xi_{n+1}$. Since $\xi\, R_g\, \xi'$, there exist now e', ξ'_{n+1} such that $\xi' \overset{e'}{\Longrightarrow} \xi'_{n+1}$, $g \cup (e,e')$ is a consistent occurrence aliasing and $\xi_{n+1}\ R_{g\cup(e,e')}\ \xi'_{n+1}$. We know from Lemma 5.8 (iii)) that $\downarrow e \subseteq \{e_1, \ldots, e_n\}$, $\downarrow e' \subseteq \{e'_1, \ldots, e'_n\}$. Then $path(e) = e_{i_1} \cdot \cdots \cdot e_{i_k}$, where $i_j \in \{1, \ldots, n\}$, $e_{i_j} \prec e_{i_{j+1}}$. Since $g \cup (e,e')$ is a consistent occurrence aliasing, we have $e'_i \prec e' \Leftrightarrow e_i \prec e$ and $e'_{i_j} \prec e'_{i_h} \Leftrightarrow e_{i_j} \prec e_{i_h}$, thus $path(e') = e'_{i_1} \cdot \cdots \cdot e'_{i_k}$. From this and $f_2(e'_i) = f_1(e_i)$, we deduce $f_2(path(e')) = f_1(path(e)) = u$. The consistency of $g \cup (e,e')$ also implies $\lambda(e') = \lambda(e) = a$. Then by Lemma 5.28 (iii) $s = proc(\xi')[f_2] \underset{ul}{\overset{a}{\Longrightarrow}}_d proc(\xi'_{n+1})\,[f_2\{l/e'\}] = s'$. Thus $(r', s') \in S$.

- Let now $r \overset{\tau}{\Longrightarrow}_d r'$. By Lemma 5.28 (ii) $\exists\, \xi_{n+1}$ such that $proc(\xi_{n+1})\,[f_1] = r'$ and $\xi \overset{\tau}{\Longrightarrow} \xi_{n+1}$. Since $\xi\, R_g\, \xi'$, there exists ξ'_{n+1} such that $\xi' \overset{\tau}{\Longrightarrow} \xi'_{n+1}$ and $\xi_{n+1}\, R_g\, \xi'_{n+1}$. By Lemma 5.28 (i) $s = proc(\xi')\,[f_2] \overset{\tau}{\Longrightarrow}_d proc(\xi'_{n+1})\,[f_2] = s'$. Since the computations $p \overset{\tau}{\Longrightarrow} \xi_0 \overset{e_1}{\Longrightarrow} \cdots \overset{e_n}{\Longrightarrow} \xi_{n+1}$ and $q \overset{\tau}{\Longrightarrow} \xi'_0 \overset{e'_1}{\Longrightarrow} \cdots \overset{e'_n}{\Longrightarrow} \xi'_{n+1}$ are of the required form, we have then $(r', s') \in S$. Again (*cf* Theorem 5.18), the first $\overset{\tau}{\Longrightarrow}$ in the computations is needed to cover the case of an initial τ-transition.

□

To prove the reverse implication, we use the alternative definition $\approx^{\nu}_{\ell}$ of dynamic location equivalence.

Theorem 5.30 *For any* $p, q \in \text{CCS}$*:* $\quad p \approx^{d}_{\ell} q \ \Rightarrow\ p \approx^{occ} q$.

PROOF: Suppose $p \approx^{\nu}_{\ell} q$. Define a $\mathcal{G}$-indexed family of relations $\mathbf{R} = \{R_g \mid g \in \mathcal{G}\}$ as follows:

$$(\xi, \xi') \in R_g \iff \text{there exist occurrence transition sequences}$$

$$p \overset{\tau}{\Longrightarrow} \xi_0 \overset{e_1}{\Longrightarrow} \cdots \overset{e_n}{\Longrightarrow} \xi_n = \xi$$

$$q \overset{\tau}{\Longrightarrow} \xi'_0 \overset{e'_1}{\Longrightarrow} \cdots \overset{e'_n}{\Longrightarrow} \xi'_n = \xi'$$

such that

$$(1)\ \ g \ = \ \{\,(e_1, e'_1), \ldots, (e_n, e'_n)\,\}$$

$$(2)\ \ \exists \text{ injection } f : \{e'_1, \ldots e'_n\} \rightarrow \mathit{Loc} \text{ such that :}$$

$$\mathit{proc}(\xi)\,[f \circ g] \ \ \approx^{\nu}_{\ell} \ \ \mathit{proc}(\xi')[f]$$

Note that $\mathit{proc}(\xi)\,[f \circ g]$, $\mathit{proc}(\xi')[f] \in \text{LCCS}^{\nu}$, given that f and g are injections. Also, it is easy to see that $(p, q) \in R_{\emptyset}$, since $\mathit{proc}(p)[\emptyset] = p$ and $\mathit{proc}(q)[\emptyset] = q$ (recall that the partial function $\emptyset$ corresponds to the identity renaming id). We show that $\mathbf{R} = \{R_g \mid g \in \mathcal{G}\}$ is a progressive o-bisimulation family. Assume $(\xi, \xi') \in R_g$. We only consider the case of observable transitions.

- Suppose $\xi \overset{e}{\Longrightarrow} \xi_{n+1}$, $e \neq \tau$. Let $l \notin \mathit{range}(f)$. Then $l \notin \supseteq \mathit{range}(f \circ g)$. By Lemma 5.28$\,(iii)$, $\mathit{proc}(\xi)\,[f \circ g] \ \underset{ul}{\overset{a}{\Longrightarrow}}_d \ \mathit{proc}(\xi_{n+1})\,[(f \circ g)\{l/e\}]$, where $a = \lambda(e)$ and $u = f \circ g(\mathit{path}(e))$. Since $\mathit{proc}(\xi)\,[f \circ g] \ \approx^{\nu}_{\ell} \ \mathit{proc}(\xi')[f]$, $\exists\, s$ s.t. $\mathit{proc}(\xi')[f] \ \underset{ul}{\overset{a}{\Longrightarrow}}_d \ s$ and $\mathit{proc}(\xi_{n+1})\,[(f \circ g)\{l/e\}] \approx^{\nu}_{\ell} s$. Now by Lemma 5.28$\,(iv)$ there exists e' s.t. $\lambda(e') = a$, $f(\mathit{path}(e')) = u$ and $\xi' \overset{e'}{\Longrightarrow} \xi'_{n+1}$ for some ξ'_{n+1} s.t. $\mathit{proc}(\xi'_{n+1})\,[f\{l/e'\}] = s$. Since $(f \circ g)\{l/e\} = f\{l/e'\} \circ (g \cup (e, e'))$, we have then $\mathit{proc}(\xi_{n+1})\,[f\{l/e'\} \circ (g \cup (e, e'))] \ \approx^{\nu}_{\ell} \ \mathit{proc}(\xi'_{n+1})\,[f\{l/e'\}]$, where $f\{l/e'\}$ is still an injection since $l \notin \mathit{range}(f)$. So all we have to show is that $g \cup (e, e')$ is a consistent occurrence aliasing. Since g is known to be one, it is enough to show that $e_i \prec e \ \Leftrightarrow \ g(e_i) \prec g(e) = e'$. But this is implied by $g(\mathit{path}(e)) = \mathit{path}(e')$, which in turn follows from $f \circ g(\mathit{path}(e)) = u = f(\mathit{path}(e'))$ and the injectivity of f. $\square$

We give now the analogous result for the preorders. To prove the direction $p \sqsubseteq\!\!\!\!\!\sim^{d}_{\ell} q \ \Rightarrow \ p \sqsubseteq\!\!\!\!\!\sim^{occ} q$ we use a preorder $\sqsubseteq\!\!\!\!\!\sim^{\nu}_{\ell}$ (the obvious variant of $\approx^{\nu}_{\ell}$) in place of $\sqsubseteq\!\!\!\!\!\sim^{d}_{\ell}$.

Theorem 5.31 *For any $p, q \in$ CCS : $\ p \sqsubseteq\!\!\!\!\!\sim^{d}_{\ell} q \ \Leftrightarrow \ p \sqsubseteq\!\!\!\!\!\sim^{occ} q$.*

PROOF: $\Leftarrow$: adapted from the proof of Theorem 5.29. We take a progressive o-prebisimulation family $\mathbf{R}$ and show that $\mathbf{S}$, as defined there, is a dynamic location prebisimulation. To show that $f_2(\mathit{path}(e'))$ is a subword of $f_1(\mathit{path}(e)) = u$, note that if $g \cup (e, e')$ is a right-consistent occurrence aliasing, then $g(e_i) \prec e' \Rightarrow e_i \prec e$ and $g(e_{i_j}) \prec g(e_{i_h}) \Rightarrow e_{i_j} \prec e_{i_h}$. But this means that $\mathit{path}(e')$ is a subword of $g(\mathit{path}(e))$, whence the result, since $f_2(g(e_i)) = f_1(e_i)$.

$\Rightarrow$: Similar adaptation of the proof of Theorem 5.30. We want to show here that $g \cup (e, e')$ is a right-consistent occurrence aliasing. It is enough to show $e'_i \prec e' \Rightarrow$

$e_i \prec e$. We know that $f(path(e'))$ is a subword of $f \circ g\,(path(e))$. Since f is injective, this implies that $path(e')$ is a subword of $g(path(e))$. It follows that $\downarrow e' \subseteq g(\downarrow e)$. Since $\downarrow e \subseteq \{e_1, \ldots, e_n\}$, $\downarrow e' \subseteq \{e'_1, \ldots, e'_n\}$, this amounts exactly to $e'_i \prec e' \Rightarrow e_i \prec e$.

□

Theorem 4.5 **(1)**, stating the coincidence of $\approx^s_\ell$ and $\approx^d_\ell$, follows now immediately from Theorems 5.18, 5.18 and 5.30, 5.30. Similarly, Theorem 4.5 **(2)** follows from the analogous results for the preorders.

Acknowledgements

The idea of a static assignment of locations was originally put forward by G. Boudol in the course of a CEDISYS meeting in Brighton, in September 1990. I would like to thank him for inspiration and for several comments and advices. I also benefitted from discussions with L. Aceto, who was the first to formalise the "static view of locations" for a subset of CCS. I am grateful to P.S. Thiagarajan and Frédéric Boussinot, for commenting on an earlier draft of this paper. Part of this work was done while visiting Thiagarajan and his colleagues in Madras. I would like to thank all of them for their interest and comments.

Appendix

We give here a definition of *local history preserving bisimulation* on the occurrence system, and show that the induced equivalence on CCS processes coincides with the equivalence $\approx^{occ}$, as was mentioned in Section 5. We also present an alternative definition of the dynamic location equivalence $\approx_\ell^d$, which is based on a *finitely branching dynamic transition system* for CCS. This is essentially the same as that proposed by Yankelevich in [Yan93], with a slightly different formulation.

Local history preserving bisimulation on CCS

Using the occurrence transition system of Section 5, we may define a notion of *local* history preserving bisimulation for CCS processes. History-preserving bisimulation was originally defined in [RT88] and [GG89] for prime event structures, and extended in [GG90] and [Ace92] to flow and stable event structures respectively. Essentially, a history-preserving bisimulation is a bisimulation which preserves, at each state of computation, the partially ordered set of events that led to that state. Our definition differs from that of [GG89] and [Ace92] in two respects: it is "syntactic", in that it is defined directly on (an enrichment of) the CCS transition system, and it is based on the local rather than the global causality ordering.

The occurrence system provides a notion of *state* (or configuration) for CCS terms. For $p \in$ CCS, define:

$$States(p) \;=\; \{\, \xi \mid \exists\, e_i, \xi_i \text{ s.t. } p \overset{\tau}{\Longrightarrow} \xi_0 \overset{e_1}{\Longrightarrow} \cdots \overset{e_n}{\Longrightarrow} \xi_n = \xi \,\}$$

Recall that each state ξ has an associated set of events $occ\,(\xi)$, ordered by the local causality relation $\preceq$. Unlike the global causality ordering in flow and stable event structures, which is relative to a configuration, the local causality ordering $\preceq$, which is essentially a static notion, is the same for all states.

Definition 5.32 (**Local history preserving bisimulation**)

Let $p, q \in$ CCS. A relation $R \subseteq States(p) \times States(q) \times \wp(\,occ\,(States(p)) \times occ\,(States(q))\,)$ is a *local history preserving bisimulation (lhp-bisimulation)* between p and q if $(p, q, \emptyset) \in R$ and whenever $(\xi_0, \xi_0', g) \in R$ then:

(1) g is an isomorphism between $(\,occ\,(\xi_0),\ \preceq\,)$ and $(\,occ\,(\xi_0'),\ \preceq\,)$

(2) *a)* $\xi_0 \overset{e}{\Longrightarrow} \xi_1 \;\Rightarrow\; \exists\, e', \xi_1'$ s.t. $\xi_0' \overset{e'}{\Longrightarrow} \xi_1'$ and $(\xi_1, \xi_1', g \cup (e, e')) \in R$

b) $\xi_0' \overset{e'}{\Longrightarrow} \xi_1' \;\Rightarrow\; \exists\, e, \xi_1$ s.t. $\xi_0 \overset{e}{\Longrightarrow} \xi_1$ and $(\xi_1, \xi_1', g \cup (e, e')) \in R$

(3) *a)* $\xi_0 \overset{\tau}{\Longrightarrow} \xi_1 \;\Rightarrow\; \exists\, \xi_1'$ s.t. $\xi_0' \overset{\tau}{\Longrightarrow} \xi_1'$ and $(\xi_1, \xi_1', g) \in R$

b) $\xi_0' \overset{\tau}{\Longrightarrow} \xi_1' \;\Rightarrow\; \exists\, \xi_1$ s.t. $\xi_0 \overset{\tau}{\Longrightarrow} \xi_1$ and $(\xi_1, \xi_1', g) \in R$

We say that p and q are *local history preserving equivalent*, $p \approx^{lhp} q$, if there exists a local history preserving bisimulation between them.

We noted already that this definition is syntactic, as opposed to the original definitions of [RT88], [GG89], which were given on event structures. By taking a slightly more concrete notion of occurrence, where communications are pairs of visible occurrences, and adopting the corresponding global causality ordering (as defined in [BC91]), we would obtain a similar syntactic definition for the usual notion of history preserving bisimulation. We have now the following:

Fact 5.33 *For any processes $p, q \in \mathrm{CCS}$: $p \approx^{lhp} q \;\Leftrightarrow\; p \approx^{occ} q$.*

Proof: It should be clear that if $\mathbf{R} = \{R_g \mid g \in \mathcal{G}\}$ is a progressive bisimulation family such that $pR_\emptyset q$, then the relation:

$$S = \{\, (\xi, \xi', g) \mid \xi \in States(p),\ \xi' \in States(q),\ g \in \mathcal{G} \ \text{ and } \ \xi R_g \xi' \,\}$$

is a lhp-bisimulation between p and q, since if $\xi R_g \xi'$ for $\xi \in States(p)$, $\xi' \in States(q)$, then g is an occurrence aliasing such that $occ(\xi) \subseteq dom(g), occ(\xi') \subseteq range(g)$, that is an isomorphism between $occ(\xi)$ and $occ(\xi')$.

Similarly, if S is a lhp-bisimulation between p and q, we define a family $\mathbf{R} = \{R_g \mid g \in \mathcal{G}\}$ by:

$$R_g = \{\, (\xi, \xi') \mid (\xi, \xi', g) \in S \,\}$$

Clearly $\mathbf{R}$ is a progressive bisimulation family such that $pR_\emptyset q$. $\square$

In the light of the results of Section 5, we have then also:

Corollary 5.34 *For any processes $p, q \in \mathrm{CCS}$: $p \approx^d_\ell q \;\Leftrightarrow\; p \approx^{lhp} q \;\Leftrightarrow\; p \approx^s_\ell q$.*

A similar notion of *local history preserving preorder*, $\sqsubseteq^{lhp}$, can be obtained by requiring g, in Definition 5.32, to be a bijection between $(\, occ(\xi_0),\ \preceq\,)$ and $(\, occ(\xi_0'),\ \preceq\,)$ whose inverse is a homomorphism.

Finitely-branching dynamic location transition system

The rest of this Appendix is devoted to showing that the infinite branching is not essential to the dynamic location transition system. As suggested in Section 4, it is possible to retrieve the equivalence $\approx^d_\ell$ by dynamically assigning a *canonical atomic location* to each transition of a CCS term. In fact, this has been shown already by Yankelevich, who introduced in [Yan93] a variant of the dynamic transition system, called *transition system with numbered localities*, where progressive natural numbers are chosen as atomic locations.

Here, with the occurrence system at our disposal, it would be natural to take as canonical atomic location for a transition the corresponding *occurrence*. More precisely, we could restrict attention to processes of the form $proc(\xi)$, where $\xi \in \mathcal{S}$ is a state in the occurrence system, and to canonical dynamic transitions of the form $proc(\xi) \xrightarrow[\sigma \cdot e]{a}_d proc(\xi')$, where e is the unique occurrence such that $\lambda(e) = a$, $path(e) = \sigma$ and $loc(e) = where(e, proc(\xi'))$ (*cf* Section 5).

It should be clear that the resulting *canonical dynamic transition system* has at most one transition for any unguarded occurrence e at each state, and thus is finitely-branching for any process with guarded recursion and finite degree of parallelism (what is called a guarded and sequential process in [Mil89]).

However, this choice of atomic locations has a drawback: to equate transitions of bisimilar processes we still have to use a bijection on their locations (although not an order-preserving one, since this is already guaranteed by the correspondence of access paths). To be able to use the definition of dynamic location bisimulation as it stands, we make a further step and consider processes $proc(\xi)[f]$, where f is an injective location renaming. In fact, we may take f to be a *monotonic injective renaming* $f : Loc(proc(\xi)) \rightarrow \{1, \ldots, n\}$, where $n = |\, Loc(proc(\xi))\,|$ (note that $Loc(proc(\xi)) = occ(\xi)$ is finite for any $\xi \in \mathcal{S}$). In this way we retrieve exactly the *transition system with numbered localities* (nl-transition system) of [Yan93].

Let us recall the definition of this nl-transition system, rephrasing it in our formalism. A location renaming $f : \mathcal{O} \rightarrow \mathbb{N}$ is monotonic if $e \prec e' \Rightarrow f(e) < f(e')$. In this case we write $f : \mathcal{O} \rightarrow_{mon} \mathbb{N}$. Assume $\mathbb{N} \subseteq Loc$.

The states of the transition system with numbered localities are:

$$Nproc = \{p \mid \exists \xi \in \mathcal{S} \text{ s.t. } p = proc(\xi)[f] \text{ and } f : occ(\xi) \rightarrow_{mon} \{1, \ldots, |\, occ(\xi)\,|\}\}$$

The transitions $p \xrightarrow[u \cdot (n+1)]{a}_{nl} p'$ on $Nproc$ are the least ones such that:

$$p \xrightarrow[u \cdot l]{a}_d p',\ l \notin Loc(p),\ \ |\, Loc(p)\,| = n \quad \Rightarrow \quad p \xrightarrow[u \cdot (n+1)]{a}_{nl} p'\{n+1/l\}$$

It should be clear that the dynamic location bisimulation equivalence based on such transitions coincides with $\approx_\ell^d$ (the proof is essentially the same as for $\approx_\ell^\nu = \approx_\ell^d$).

Again, this transition system is finitely-branching for processes with guarded recursion and finite degree of parallelism. However, the nl-transition system is "wider" than the dynamic canonical transition system, since the diamonds corresponding to concurrency are unfolded, thus giving rise to duplication of states. This is due to the fact that for any CCS term p, the nl-transition system assigns location n to the *nth* transition of any computation of p, hence the order in which concurrent transitions are executed is recorded in the states.

The difference between the various transition systems is illustrated by the following example, discussed already in [MN92].

Example 5.35 Consider the CCS process $p = rec\,x.\ a.x \mid rec\,x.\ a.x$. Its standard transition system has just one state p and one transition $p \xrightarrow{a} p$, while its static canonical transition system has one state $p' = 0 :: rec\,x.\ a.x \mid 1 :: rec\,x.\ a.x$ and two transitions $p' \xrightarrow[0]{a}_s p'$ and $p' \xrightarrow[1]{a}_s p'$. On the other hand both the dynamic canonical transition system and the transition system with numbered localities are infinite (although finitely-branching), as illustrated by the figure below. This is because the dynamic location transition systems, as well as the occurrence transition system of Section 5, are models of computation rather than system models.

The dynamic canonical transition system for p is the following:

$$\begin{tikzcd}
& p \arrow[dl, "a"', "0"] \arrow[dr, "a", "1"'] & \\
0a :: rec\, x.\ a.x \mid rec\, x.\ a.x \arrow[dr, "a", "1"'] & & rec\, x.\ a.x \mid 1a :: rec\, x.\ a.x \arrow[dl, "a"', "0"] \\
& 0a :: rec\, x.\ a.x \mid 1a :: rec\, x.\ a.x \arrow[dl] \arrow[dr] & \\
{} & \cdots & {}
\end{tikzcd}$$

while the transition system with numbered localities is:

$$\begin{tikzcd}
& p \arrow[dl, "a"', "0"] \arrow[dr, "a", "0"'] & \\
0 :: rec\, x.\ a.x \mid rec\, x.\ a.x \arrow[d, "a"', "1"] & & rec\, x.\ a.x \mid 0 :: rec\, x.\ a.x \arrow[d, "a"', "1"] \\
0 :: rec\, x.\ a.x \mid 1 :: rec\, x.\ a.x \arrow[d, "\cdots", shift left] \arrow[d, shift right] & & 1 :: rec\, x.\ a.x \mid 0 :: rec\, x.\ a.x \arrow[d, "\cdots", shift left] \arrow[d, shift right] \\
{} & & {}
\end{tikzcd}$$

Note the unfolding of concurrency diamonds here.

References

[Ace91] L. Aceto. A static view of localities. Report 1483, INRIA, 1991. To appear in *Formal Aspects of Computing*.

[Ace92] L. Aceto. History preserving, causal and mixed-ordering equivalence over stable event structures (note). *Fundamenta Informaticae*, 17(4), 1992.

[BC91] G. Boudol and I. Castellani. Flow models of distributed computations: three equivalent semantics for CCS. Report 1484, INRIA, 1991. To appear in *Information and Computation*. Preliminary version in *Proceedings LITP Spring School*, La Roche-Posay, number 469 in LNCS, 1990.

[BCHK91] G. Boudol, I. Castellani, M. Hennessy, and A. Kiehn. A theory of processes with localities. Report 1632, INRIA, 1991. To appear in *Formal Aspects of Computing*. Extended abstract in *Proceedings CONCUR92*, number 630 in LNCS, 1992.

[BCHK93] G. Boudol, I. Castellani, M. Hennessy, and A. Kiehn. Observing localities. *TCS*, 114:31–61, 1993. Extended abstract in *Proceedings MFCS 91*, number 520 in LNCS, 1991.

[Cas88] I. Castellani. *Bisimulations for Concurrency*. Ph.d. thesis, University of Edinburgh, 1988.

[Cas93] I. Castellani. Observing distribution in processes. In *Proceedings MFCS 93*, number 711 in LNCS, 1993.

[CH89] I. Castellani and M. Hennessy. Distributed bisimulations. *JACM*, 10(4):887–911, 1989.

[Chr92] S. Christensen. Distributed bisimilarity is decidable for a class of infinite state-space systems. In *Proceedings CONCUR 92*, number 630 in LNCS, 1992.

[CN93] F. Corradini and R. De Nicola. Locality and causality in distributed process algebra. Report SI/RR - 93/05, Università di Roma La Sapienza, 1993.

[DD90] Ph. Darondeau and P. Degano. Causal trees: interleaving + causality. In *Proceedings LITP Spring School,* La Roche-Posay, number 469 in LNCS, 1990.

[DD91] Ph. Darondeau and P. Degano. Event structures, causal trees and refinement, 1991. Submitted to Theoretical Computer Science for the special issue of MFCS 90.

[DDNM89] P. Degano, R. De Nicola, and U. Montanari. Partial orderings descriptions and observations of nondeterministic concurrent processes. In *Proceedings REX School/Workshop,* Noordwijkerhout, number 354 in LNCS, 1989.

[GG89] R.J. van Glabbeek and U. Goltz. Equivalence notions for concurrent systems and refinement of actions. GMD Technical Report 366, Gesellschaft für Mathematik und Datenverarbeitung, Sankt Augustin, 1989. Extended abstract in *Proceedings MFCS 89*, number 379 in LNCS, 1989.

[GG90] R.J. van Glabbeek and U. Goltz. Equivalences and refinement. In *Proceedings LITP Spring School*, number 469 in LNCS, 1990.

[Kie89] A. Kiehn. Distributed bisimulations for finite CCS. Report 7/89, University of Sussex, 1989.

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

[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, 1989.

[MN92] M. Mukund and M. Nielsen. CCS, locations and asynchronous transition systems. In *Proceedings FST-TCS 92*, number 652 in LNCS, 1992.

[Mur93] D. Murphy. Observing located concurrency. In *Proceedings MFCS 93*, number 711 in LNCS, 1993.

[MY92] U. Montanari and D. Yankelevich. A parametric approach to localities. In *Proceedings ICALP 92*, number 623 in LNCS, 1992.

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

[Yan93] D. Yankelevich. *Parametric Views of Process Description Languages.* Ph.d. thesis, University of Pisa, 1993.

Éditeur
INRIA, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex (France)
ISSN 0249-6399