Observing Localities *
(Extended Abstract)

G. Boudol, I. Castellani
INRIA, Sophia-Antipolis,
M. Hennessy
CSAI, University of Sussex,
A. Kiehn
TUM, Munich.

1 Introduction

There are by now a number of well-established semantic theories of processes in the research literature which are based on principles of observation. The main idea is that processes are deemed to be equivalent if there is no possible observation which can distinguish them. Different formalisations of this idea, which give rise to a number of semantic equivalences, may be found in [Mil89], [Hen88] and [Hoa85]. All of these formalisations are based on the same simple notion of observation, namely communication: one may observe a process by communicating with it via a communication channel. The resulting semantic theories are often called *interleaving theories*; they do not distinguish between concurrency and nondeterminism or more precisely they equate a parallel process with the purely nondeterministic one obtained by interleaving its primitive computation steps or actions.

Some attempts have been made to generalise this observation based approach in order to develop a semantic theory which does distinguish between these two phenomena, [CH89], [BC87], [DNM88], [BC89], [DD89]. Here we reexamine the approach taken in [CH89] and [Kie89] where the processes under observation are considered to be *distributed in nature*. So the observer can not only test the process by communicating with it but can also observe or distinguish that part of the distributed process which reacted to the test. A purely nondeterministic process is based at one site whereas in general a concurrent one may be distributed among a number of different locations. It follows that an observer will be able to distinguish them.

*This work has been supported by the ESPRIT/BRA CEDISYS project.

We use as a starting point the process algebra CCS, a process description language which describes processes in terms of the *actions* they can perform. So for example the process B_1 defined by

$$B_1 \Leftarrow in.out.B_1$$

is a simple process which repeatedly performs the actions *in, out.* If we run two copies of this in parallel we obtain a process which acts more or less like a two-place bag:

$$B_2 \Leftarrow B_1 \mid B_1.$$

Here $\mid$ is the parallel operator of CCS which in this context defines a process which consists of two independent processes, two copies of B_1, running in parallel.

Processes running in parallel may also communicate or synchronise with each other. This is formalised by dividing the set of actions into two complementary subsets, the *input* actions and the *output* actions. Communication is then considered to be the simultaneous occurrence of complementary actions. Output actions are indicated by an overbar, such as $\overline{a}, \overline{in}$, etc., input actions by the absence of an overbar and there is a distinguished action τ to indicate a communication or more generally internal and unobservable computation. So if we define two processes *In, Out* by

$$\begin{aligned} In &\Leftarrow in.\alpha.In \\ Out &\Leftarrow \overline{\alpha}.out.Out \end{aligned}$$

then the process $In \mid Out$ acts somewhat like B_2. However *In, Out* are not obliged to synchronise via the action α. The actions α and $\overline{\alpha}$ may be performed independently, which corresponds to separate synchronisations with processes in their operating environments. To eliminate these possible communications with the environment and thereby force the synchronisation between the two processes we limit the scope of these actions using another operator of CCS, *restriction*, which in general is written as $\backslash A$ where A is a set of actions. So let NB_2 be defined by

$$NB_2 \Leftarrow (In \mid Out)\backslash\alpha.$$

These two processes, B_2 and NB_2, offer very similar behaviour to a user particularly as the synchronisation between *In* and *Out* is not supposed to be visible externally. According to the theory developed in [Mil89] they are *weak bisimulation equivalent*, denoted by $B_2 \approx NB_2$; in terms of the visible actions *in* and *out* they offer the same possible behaviour to any user of the systems.

However this reasoning is based on the assumption that the only property which can be observed of a process is its ability to perform particular actions. Now let us reinterpret the language by saying that $P \mid Q$ is a *distributed process* where the sub-process

P is at one site and Q is at another site; moreover let us suppose that an observer can distinguish between sites in the sense that when a distributed process performs an action the observer knows the site responsible for it. Thus one observer's view of the distributed process B_2 is

Here the observer has decided, out of personal choice, to call the site or location of the first subprocess l_1 and the second subprocess l_2. Now it is not possible to construct a similar view of NB_2. For example

can easily be distinguished as here all *in* actions are seen to occur at the location l_1 and all *out* actions at location l_2. In contrast they are distributed between l_1 and l_2 in the distributed process B_2.

The basic difference between these two processes is that in NB_2 one site is responsible for the *in* actions and one for the *out* actions whereas B_2 has two equivalent sites each acting like a one place buffer. Viewing these as specifications this is a useful and meaningful distinction. To implement B_2 it is necessary to have independent locations each acting like buffers whereas an implementation of NB_2 would always have to localise the responsibility for the *in* actions and the *out* actions in independent locations.

Let us now address the question of how exactly an observer should be allowed to perceive the distributed nature of a process. In this respect we are guided by principles of extensionality; we would like the resulting equivalence to be as extensional as possible in that the semantics of a process should only be determined by its external manifestations rather than its internal structure or behaviour. It is reasonable to argue that at least some aspect of the distribution of subprocesses of a distributed system is a part of its extensional behaviour and therefore if we are to view CCS as a language for describing distributed systems an observer should be able to view in $P \mid Q$ a distributed system which potentially has two distinct sites; any externally visible action performed by P should be recognizable as emanating from one location and any performed by Q should be recognizable as coming from a different location.

In this paper we develop the idea that location names are assigned dynamically as

part and parcel of the act of observation; when an observer sees an action it will naturally be seen as emanating from a particular location and the observer may then choose to allocate a name to that location. All subsequent actions performed at that location will then be recognised as emanating from this same location. Technically this involves developing an operational semantics for the language by replacing the usual statements of the form $P \xrightarrow{a} Q$ with new ones of the form

$$P \xrightarrow[u]{a} Q$$

which carry information about the location names which the observer has assigned to particular locations in the system. This is very similar to the approach taken in [CH89], [Cas88] and [Kie89] where *distributed bisimulations* are defined and in the full version of the paper we offer a detailed comparison.

2 Location Equivalence

The language we use to formalize our approach is a slight extension of Milner's pure CCS. The main extension is an additional operator called *location prefixing* representing the allocation of locations to processes. A process p prefixed by a location u will be denoted by $u :: p$. Intuitively, this means that process p is at a location called u. However, in general we will assume these locations are introduced via the observation of visible actions. That is, initially, before any experiment has been performed the process under investigation does not contain any location. With the observation of an action the location of the action is also perceived and assigned a name. Thus the observers we assume here are more powerful than those usually considered for CCS or other process algebras. So we will have a transition rule

$$a.p \xrightarrow[u]{a} u :: p \quad \text{for any location name } u$$

which means that a is performed at a location to which the observer permanently assigns the name u. If further experiments are performed on $u :: p$ then the location u will always be observed. Moreover the location called u may contain sub-locations which in turn may also be observed. For example

$$u :: (a.nil \mid b.nil) \xrightarrow[uv]{a} u :: (v :: nil \mid b.nil).$$

Here the location which has been called u by a previous observation contains two sub-locations and at one of them a is performed. The name v is allocated to this subsite via the observation of a.

Formally we assume an infinite set of basic location names or site names Loc, ranged over by $l, k, etc.$, and general locations will be sequences from Loc^*, ranged over by

$u, v, w, \ldots$. So we work with the following abstract syntax:

$$t \;=\; nil \;\mid\; \mu.t \;\mid\; t+t \;\mid\; t \mid t \;\mid\; t[f] \;\mid\; t\backslash b$$
$$\mid\; x \;\mid\; rec x.\, t \;\mid\; t \between t \;\mid\; t \mid_c t \;\mid\; u :: t$$

where μ ranges over $Act \cup \{\tau\}$, Act is a basic set of actions and f is a bijective relabelling function which preserves complementation. Most of these operators are taken directly from CCS but $\between$ and $\mid_c$ are the leftmerge and communication merge operators from [BK85], [Hen88].

We will give two operational semantics to the closed terms of this language. The first one generalizes bisimulation equivalence, $\approx$, in a straightforward way and is omitted in this extended abstract. This equivalence considers the ability of performing visible actions and only in this respect bisimilar processes exhibit the same behaviour. The second semantics we give to the language additionally takes the distribution in space into account. As already discussed above there will be two points in which it differs from bisimulation equivalence. The first point is that locations may be introduced via the observation of actions. Secondly processes may contain locations and actions in the scope of locations will be observed at those locations. Formally these two points are reflected by the following two rules in the location transition system.

(L1) $\qquad a.p \xrightarrow[u]{a} u :: p \qquad u \in Loc^*$

(L2) $\qquad p \xrightarrow[u]{a} p' \quad$ implies $\quad v :: p \xrightarrow[vu]{a} v :: p'$

Here u is an access path representing a location where the action a is performed and in the second rule this is extended by v to give the new location vu.

The τ–transitions are considered, as usual, to be invisible, so no location is observed when they are performed. They are of the form $p \xrightarrow{\tau} p'$ and are defined through the standard transition system for CCS on which bisimulation equivalence is based. Visible transitions are defined by the location transition system given in Figure 1. They always have the form $p \xrightarrow[u]{a} p'$ where we call u the location where the action a is performed. Weak transitions are defined in the usual manner:

$$p \stackrel{\varepsilon}{\Rightarrow} p_1,\; p_1 \xrightarrow[u]{a} p_2,\; p_2 \stackrel{\varepsilon}{\Rightarrow} p' \quad \text{implies} \quad p \stackrel{a}{\underset{u}{\Longrightarrow}} p'.$$

We can now see how parallelism is differentiated from nondeterminism. For the process $a.nil \mid b.nil$ we can derive $a.nil \mid b.nil \xrightarrow[l]{a} l :: nil \mid b.nil$ while its nondeterministic counterpart would perform the transition $a.b.nil + b.a.nil \xrightarrow[l]{a} l :: b.nil$. Now with the observation of the action b different locations would be perceived. In $l :: nil \mid b.nil \xrightarrow[k]{b} l :: nil \mid k :: nil$ the b is performed at the location k which is independent of l whereas in $l :: b.nil \xrightarrow[lk]{b} lk :: nil$ it is performed at a sublocation of l, namely lk.

For each $a \in Act$ let $\xrightarrow[u]{a} \subseteq (\mathbb{P} \times Loc^* \times \mathbb{P})$ be the least binary relation satisfying the following axioms and rules.

(L1)	$a.p \xrightarrow[u]{a} u :: p$	$u \in Loc^*$	
(L2)	$p \xrightarrow[u]{a} p'$	implies	$v :: p \xrightarrow[vu]{a} v :: p'$
(L3)	$p \xrightarrow[u]{a} p'$	implies	$p + q \xrightarrow[u]{a} p'$ $q + p \xrightarrow[u]{a} p'$
(L4)	$p \xrightarrow[u]{a} p'$	implies	$p \mid q \xrightarrow[u]{a} p' \mid q$ $q \mid p \xrightarrow[u]{a} q \mid p'$
(L5)	$p \xrightarrow[u]{a} p'$	implies	$p \between q \xrightarrow[u]{a} p' \mid q$
(L6)	$p \xrightarrow[u]{a} p'$	implies	$p[f] \xrightarrow[u]{f(a)} p'[f]$
(L7)	$p \xrightarrow[u]{a} p'$	implies	$p\backslash b \xrightarrow[u]{a} p'\backslash b,\ a \notin \{b, \bar{b}\}$
(L8)	$t[rec\,x.\ t/x] \xrightarrow[u]{a} p'$	implies	$rec\,x.\ t \xrightarrow[u]{a} p'$

Figure 1: Location Transitions

Based on this transition system we now define location equivalence. Two processes p and q are location equivalent if every move of one of them is matched by a similar move of the other and in particular if for every visible transition $p \overset{a}{\underset{u}{\Longrightarrow}} p'$ the matching transition $q \overset{a}{\underset{u}{\Longrightarrow}} q'$ has the same location.

Definition 2.1 [Location Equivalence]
A symmetric relation $R \subseteq \mathbb{P} \times \mathbb{P}$ is called a *location bisimulation* iff $R \subseteq C(R)$ where $(p, q) \in C(R)$ iff

(i) $p \overset{\varepsilon}{\Rightarrow} p'$ implies $q \overset{\varepsilon}{\Rightarrow} q'$ for some $q' \in \mathbb{P}$ such that $(p', q') \in R$

(ii) $p \overset{a}{\underset{u}{\Longrightarrow}} p', a \in Act, u \in Loc^*$

implies $q \overset{a}{\underset{u}{\Longrightarrow}} q'$ for some $q' \in \mathbb{P}$ such that $(p', q') \in R$.

Two processes p and q are said to be *location equivalent*, $p \approx_l q$, iff there is a location bisimulation R such that $(p, q) \in R$. □

One may now easily check that the two processes B_2 and NB_2 defined in the introduction, are distinguished by $\approx_l$.

3 Properties of Location Equivalence

In this section we briefly outline some properties of $\approx_l$. The details may be found in the full version of the paper.

Proposition 3.1

1. $p \approx_l q$ *implies* $p \approx q$.
2. *If* p, q *contain no occurrence of* $|$ *then* $p \approx q$ *implies* $p \approx_l q$.
3. *Let* $p, q \in \mathbb{P}$, *and* $\pi: Loc \longmapsto Loc^*$ *be an arbitrary relabelling of location names. Then* $p \approx_l q$ *implies* $\pi(p) \approx_l \pi(q)$.
4. $\approx_l$ *is preserved by all the operators of* CCS *except* $+$ *and* $\not|$.

Like weak bisimulation $\approx$, the equivalence $\approx_l$ is not preserved by $+$. Let $\approx_l^c$ be the largest equivalence contained in $\approx_l$ which is preserved by all operators. This congruence satisfies the usual static laws of CCS [Mil89] and may be characterised equationally on the subset of CCS which does not include restriction and relabelling. The required equations, which involve the auxiliary operators $\not|$, $|_c$ are given in Figure 2, where the notation $x \sqsubset y$ means that x is absorbed by y, i. e. $y + x = y$. In the full paper we show that on the same sub-language location equivalence $\approx_l$ coincides with *distributed bisimulation* as defined in [CH89], [Kie89].

It is well known that bisimulation equivalence may be characterized using a simple modal language called *HML*, in the sense that two processes are bisimulation equivalent if and only if they satisfy exactly the same set of formulae, [Mil89]. *HML* is a simple modal logic based on two modalities $\langle a \rangle$ and $[a]$ where a is an arbitrary action. So an obvious extension to cope with location equivalence is to parameterise these modalities by locations. However, we will introduce a slightly more general modal language, which we feel is somewhat more natural. If a process contains no locations then it should be unnecessary for the formulae which characterise it to contain locations. In such processes, elements of CCS, locations are potential rather than actual and it should also be thus in their characterising formulae. For this reason we introduce location variables and quantification over these variables. One can imagine an expressive term language for locations but here we consider only a very simple language given by

$$t ::= l, l \in Loc \quad | \quad x, x \in LVar \quad | \quad \varepsilon \quad | \quad t.t.$$

$$
\begin{array}{lrcl}
\text{(A1)} & x + (y + z) & = & (x + y) + z \\
\text{(A2)} & x + y & = & y + x \\
\text{(A3)} & x + nil & = & x \\
\text{(A4)} & x + x & = & x \\
 & & & \\
\text{(LP1)} & (x + y) \between z & = & x \between z + y \between z \\
\text{(LP2)} & (x \between y) \between z & = & x \between (y \mid z) \\
\text{(LP3)} & x \between nil & = & x \\
\text{(LP4)} & nil \between x & = & nil \\
 & & & \\
\text{(I1)} & x + \tau x & = & \tau.x \\
\text{(I2)} & \mu.\tau x & = & \mu.x \\
\text{(I3)} & \mu.(x + \tau.y) + \mu.y & = & \mu.(x + \tau.y) \\
 & & & \\
\text{(NI1)} & \tau.x \between y & = & \tau.(x \mid y) \\
\text{(NI2)} & x \between y & = & x \between \tau.y \\
\text{(NI3)} & x \between (y + \tau.z) + x \between z & = & x \between (y + \tau.z) \\
 & & & \\
\text{(CPE)} & x \mid y & = & x \between y + y \between x + x \mid_c y \\
 & & & \\
\text{(CP0)} & \tau.x \mid_c y & = & x \mid_c y \\
\text{(CP1)} & (x + y) \mid_c z & = & (x \mid_c z) + (y \mid_c z) \\
\text{(CP2)} & x \mid_c y & = & y \mid_c x \\
\text{(CP3)} & x \mid_c nil & = & nil \\
\text{(CP4)} & (a.x \between x') \mid_c (b.y \between y') & = & \begin{cases} \tau.(x \mid y) \between (x' \mid y') & \text{if } a = \bar{b} \\ nil & \text{otherwise} \end{cases} \\
\text{(CP5)} & a.(x \mid x') \between (y \mid y') & \sqsubset & a.(c.x \between x' + v) \between (\bar{c}y \between y' + w)
\end{array}
$$

Figure 2: Equations G: Standard Non-Interleaving Laws.

Here *LVar* is a set of location variables, ε represents the degenerate location and is sequence concatenation. The language for property formulae is then defined by the following abstract syntax

$$\begin{array}{rcl} \Phi & ::= & \bigwedge\{\Phi \mid \Phi \in I\} \quad \mid \neg\Phi \\ & & \langle a\rangle_t\Phi \quad \mid \quad \langle\varepsilon\rangle\Phi \\ & & t = t' \quad \mid \quad \exists x.\Phi \end{array}$$

In the formula $\langle a\rangle_t\Phi$ the term t represents a location and intuitively the formula is satisfied by a process which can perform an action a at a location specified by t and in so doing reaches a state which satisfies Φ. Both $\forall$ and $\exists$ bind location variables and this leads to the usual definition of free and bound variables. We are only interested in formulae which are closed , i.e. which contain no free occurrences of location variables, which we denote by $\mathcal{L}$. More generally, for $L \subseteq Loc$ we let $\mathcal{L}_L$ denote the set of closed formulae which only use locations from L. So in particular formulae in $\mathcal{L}_\emptyset$ use no locations and all location variables are bound.

The satisfaction relation between processes and formulae, $\models \subseteq \mathbb{P} \times \mathcal{L}$, is a straightforward extension of the standard one, [Mil89], and is defined by structural induction on formulae. For example

$$\begin{array}{ll} p \models \langle a\rangle_u\Phi & \text{if } p \stackrel{a}{\Longrightarrow}_u p' \text{ and } p' \models \Phi \\ p \models \exists x.\Phi & \text{if for some } u \in Loc^* \;\; p \models \Phi[x \to u]. \end{array}$$

Theorem 3.2 *$p \approx_l q$ iff $\mathcal{L}_L(p) = \mathcal{L}_L(q)$ where $L = loc(p) \cup loc(q)$.*

As an immediate corollary we have

Corollary 3.3 *For $p, q \in$ CCS, $p \approx_l q$ if and only if $\mathcal{L}_\emptyset(p) = \mathcal{L}_\emptyset(q)$.* □

These technical results, only sketched here, show that location equivalence is a natural refinement of bisimulation equivalence which inherits many of its interesting properties but distinguishes between nondeterminism and concurrency.

References

[BC87] G. Boudol and I. Castellani. On the semantics of concurrency: partial orders and transition systems. In *Proceedings of TAPSOFT 87*, number 249 in Lecture Notes in Computer Science, pages 123–137, 1987.

[BC89] G. Boudol and I. Castellani. Permutation of transitions: an event structure semantics for CCS and SCCS. In *Proceedings of Linear Time, Branching Time and Partial Order in Logics and Models for Concurrency*, number 354 in Lecture Notes in Computer Science, pages 411–427, 1989.

[BK85] J. Bergstra and J.W. Klop. Algebra of communicating processes with abstraction. *Theoretical Computer Science*, (37):77–121, 1985.

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

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

[DD89] P. Degano and P. Darondeau. Causal trees. In *Proceedings of ICALP 88*, number 372 in Lecture Notes in Computer Science, pages 234–248, 1989.

[DNM88] P. Degano, R. De Nicola, and U. Montanari. A distributed operational semantics for CCS based on condition/event systems. *Acta Informatica*, 26:59–91, 1988.

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

[Hoa85] C.A.R. Hoare. *Communicating Sequential Processes.* Prentice-Hall, 1985.

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

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