Theoretical Computer Science 101 (1992) 223-237
Elsevier

Experimenting with process equivalence

Bard Bloom*
Department of Mathematics and Computer Science, Cornell University, Ithaca, NY 14853, USA

Albert R. Meyer**
MIT Laboratory for Computer Science, 545 Technology Square, Cambridge, MA 02139, USA

Abstract

Bloom, B. and A.R. Meyer, Experimenting with process equivalence, Theoretical Computer Science 101 (1992) 223-237.

Distinctions between concurrent processes based on observable outcomes of computational experiments are examined. The equivalence determined by a general class of experiments involving duplication of processes can be characterized by a notion of *ready simulation* resembling, but strictly coarser than, Milner's bisimulation equivalence.

1. Introduction

In the "interleaving" approach to concurrent process theory, the operational behavior of a process is completely captured by a *synchronization tree*, a rooted, unordered tree whose edges are labelled with symbols denoting basic *actions* or events, which is generally infinite and nondeterministic [14, 17, 10, 12, 3, 5, 9, 4, 11, 22]. Milner's CCS [17, 19], Hoare's CSP [14, 15] and Hennessy's algebraic theory of processes [12] are notable theories of this kind.

The concept of an internal "hidden" or τ-action is another important aspect and point of contrast among these theories. However, in this paper we restrict ourselves to the technically simpler case without internal τ-actions. We expect many of our results to generalize to the case with internal actions, but we have not investigated this as yet.

These interleaving theories further agree that synchronization trees are an over-specification of process behavior—certain distinct trees must be regarded as equivalent processes. The main theoretical difference among the theories is in which trees are identified.

* Supported by an NSF Fellowship, also NSF Grants No. 8511190-DCR and No. CCR-9003441 and ONR grant No. N00014-83-K-0125.
** Supported by an NSF Grant No. 8511190-DCR and by ONR grant No. N00014-83-K-0125.

Elsevier Science Publishers B.V.

In CSP [15], two processes are identified if they are equivalent with respect to a limited class of "button-pushing" experiments. A process is thought of as a black box, with one button for each action it can take. The experimenter presses buttons on the box. If the process can actually take that action, the machine will change state; if it cannot, the button does not allow itself to be pressed. Two processes are identified if they can perform the same sequences of actions followed by the same set of failures. An independent notion of *experimental equivalence* is defined by De Nicola and Hennessy [10] considering both necessary and possible success of certain interactive experiments on processes. In our setting without hidden moves, De Nicola-Hennessy experimental equivalence and the CSP failure-experiment equivalence coincide [12] (see also [2] for an extensive algebraic analysis of a variety of testing scenarios).

CCS is based on a finer equivalence relation on synchronization trees called *bisimulation* [19]. Although Milner's original definition of bisimulation was not given in terms of button-pushing experiments on black boxes, he does offer a justification in these terms in [18]. In these experiments, the experimenter is given the ability to perform repeated subexperiments from any state, allowing the exploration of the alternatives available in a given state. This may be phrased in several ways; for example, one might permit the experimenter to *save* states and later *restore* the process to any saved state. These must be the only operations on states, e.g., the experimenter cannot test states for equality. An alternative formulation is that the experimenter is equipped with a *duplicator*, allowing the creation of identical copies of the process in any state. The experimenter may perform experiments on the copies, and combine the results.

In general, an experiment on a process P should consist of placing P in a context, $C[P]$, involving other processes, and performing experiments on $C[P]$. However, it will turn out that the use of contexts expressible in CCS—or indeed in a very generous class of extensions of CCS—does not change any of the experimental equivalences which we consider, and so it suffices simply to perform experiments on isolated processes. In other words, all the experimental equivalences mentioned above and considered below are in fact congruences.

In the next section, we offer what we consider to be the most natural formalization of the kind of experiments with copying described informally above. We call these *duplicator* experiments. Our first observation is that despite similar motivation and informed description, duplicator experiments differ notably from the experiments which characterize bisimulation [18].

Proposition 1.1. *Equivalence with respect to duplicator experiments is a strictly coarser relation than bisimulation.*

One of the main results of this paper is that duplicator experimental equivalence coincides with *GSOS congruence* [7]. GSOS congruence was originally proposed by Istrail and the authors as a formulation of the finest relation distinguishing

processes by the completed action sequences visible when the processes appeared in contexts whose behavior had a structured operational semantics (SOS) specification [23]. It was argued in [7] that equivalences such as bisimulation which were strictly finer than GSOS congruence had questionable justification from a computational viewpoint.

GSOS congruence has two further formulations resembling the original definition of bisimulation and the subsequent characterization of bisimulation by formulas of Hennessy-Milner logic (cf. Section 4). The principal definitions are summarized in the following theorem. The equivalence of the first two definitions is one new result of this paper.

Definition 1.2 [16]. A *ready simulation*[1] is a relation $\trianglelefteq$ between processes such that if $P \trianglelefteq Q$, for each action a

- If $P \xrightarrow{a} P'$, then $Q \xrightarrow{a} Q'$ for some Q' with $P' \trianglelefteq Q'$;
- If $P \xrightarrow{a}$ then $Q \xrightarrow{a}$.

$P \sqsubseteq Q$ iff there is a ready simulation $\trianglelefteq$ such that $P \trianglelefteq Q$. P and Q are *ready* similar, $P \rightleftharpoons Q$, iff $P \sqsubseteq Q \sqsubseteq P$.

Theorem 1.3. *The following equivalence relations on synchronization trees are the same*:

- *ready similarity* [16],
- *duplicator-test equivalence*,
- *GSOS congruence* [7],
- *denial formula equivalence* [7].

The crucial difference between our duplicator experiments and Milner's is that Milner's are able to do more than make duplicates of a process. A Milner experimenter is able to *know* when enough duplicates have been made to explore all possible alternative behaviors of the duplicated process. Milner uses the metaphor of "weather conditions"—these determine which nondeterministic choice the process will make. The experimenter is allowed to choose the sequence of weather conditions that the process will experience [18]. This gives the experimenter the ability to observe the process in all possible nondeterministic behaviors.

It is clear that, in this scenario, the number of weathers available must vary over the course of the experiment, although (as the experimenter is expected to test the process in all weathers, and we require experiments to take finite time) it must always be finite in each state. If the number of weathers were fixed (say, at k), then it would not be possible to explore all the behaviors of a process capable of $k+1$-fold branching. The number of weathers cannot even be fixed for a particular process; it is straightforward to write in CCS a process with at least n-fold branching on its nth step.

[1] Originally called $\frac{2}{3}$-*bisimulation* in [16].

Milner's weather experimenter has access to overly detailed information about the process' synchronization tree. As a consequence, data is available to enable the weather experimenter to distinguish, for example, $aa + ab$ and $ab + aa$.[2] Our notion of global-testing duplicator experiments first arose from our effort to formalize an informal explanation given to us by Milner, and he agrees that our formulation clearly expresses his intentions [20].

We prefer to think of the ability to observe all nondeterministic behaviors and know that they have all been seen as a component of the testing system in its own right. Following Abramsky [1], we equip the experimenter with a *global-testing duplicator* with precisely this capability. However, allowing the experimenter unrestricted use of global-testing outcomes still disagrees with Milner's scenario, as follows from the next theorem.

Theorem 1.4. *Two processes are equivalent with respect to global-testing experiments iff they are* isomorphic *as unordered labelled trees. In particular, equivalence with respect to global-testing experiments is strictly finer than bisimulation.*

To arrive at bisimulation in the "weather" setting, Milner implicitly places a restriction on the experimenter: he is forbidden to *count* the number of kinds of weather available. The experimenter can collect results of experiments on duplicate processes only by asking whether *all* or *any* experiments succeed, not how many.

In [20], Milner refined the metaphor: one imagines continuously turning a dial, each setting of which determines a weather. The dial has no markings, and it is impossible to tell which weather is active at a given instant. An experiment consists of the usual kinds of button-pushing, and a turn-the-dial experiment involving performing a subexperiment in each of the (infinitely many) settings of the dial, and taking the conjunction or disjunction of the results. Call these *modal* global-testing experiments; we can now rephrase Milner's result as follows.

Theorem 1.5 (Milner [18]). *Two processes are bisimilar iff they agree on all* modal *global-testing duplicator experiments.*

Approximation between processes has been usefully interpreted in the Hoare and Hennessy process theories as a satisfaction relation between an implementing process and a less determinate specifying process; this has lead to a methodology for process specification and verification. Ready simulation is a notion of process approximation which may likewise be interpreted as specification satisfaction. In fact, there are straightforward notions of approximation associated with each of the alternative definitions of ready simulation in Theorem 1.3 above, all of which coincide for

[2] If s_1 and s_2 are synchronization trees, then their sum $s_1 + s_2$ is the tree obtained by identifying the roots of s_1 and s_2.

finitely branching processes. We note that, for processes without internal actions, no characterizations of bisimulation in terms of an approximation relation is known.

2. Experiments on machines

In general, we wish to justify a notion of process equivalence $P \simeq Q$ by giving an experimental scenario for it. That is, $P \not\simeq Q$ if and only if there is some experiment which distinguishes them. Scenarios serve as plausibility arguments for notions of process equivalence; they may be used to judge the appropriateness of particular equivalences. If a particularly appealing scenario precisely captures $\simeq$ (as the button-pushing scenarios described below precisely capture refusal semantics) then $\simeq$ becomes more appealing. Conversely, a mathematically beautiful notion of process equivalence may be called into question by the nonexistence of a plausible testing scenario for it. As "plausible testing scenario" is an informal notion, this criterion cannot be used to refute any process equivalence absolutely.

The idea of "button-pushing" experiments on processes has been highlighted by Hoare as an explanation of CSP semantics. A process is presented as a black box with buttons labelled with the visible atomic actions, and no other controls. If process P can perform action a, then it is possible to press the a-button and then the machine will change state. If P cannot perform a, then the a-button is locked; the experimenter can press the button, discover that the machine cannot perform an a, and then continue experimenting on P itself.

A number of variants of simple button-pushing experiments have been considered [24, 21]. Perhaps the most detailed kind of simple button-pushing is a *lighted-button experiment.* In this scenario, the black box resembles certain soft-drink machines: its buttons have lights inside them, and the light on the a-button is lit when that button is disabled. In other words, the experimenter can see at every stage which actions are possible and which are not, without changing the state of the machine. Formally, a lighted-button experiment is a sequence $S_0 a_0 S_1 \ldots a_n S_{n+1}$ alternating between sets S_i of actions and actions $a_i \notin S_i$; it succeeds when S_0 is the set of initially disabled actions, S_1 is the set of disabled actions after a_1 is pressed, and so on.

Definition 2.1. Two processes are *equivalent* with respect to a class of experiments iff they can succeed on precisely the same experiments of that class.

Both CSP equivalence and Phillips' somewhat finer *refusal testing* equivalence [22] can be characterized as equivalences based on slight restrictions of lighted-button experiments.

Note that a process may be able both to succeed and to fail on the same experiment, depending on which nondeterministic choices the process makes. For example, the process $ab + ac$ can pass the experiment $\{b, c\}a\{a, c\}$ if it takes its first a-action

alternative or fail if it takes its second alternative. On the other hand, the process $a(b+c)$ cannot pass this experiment, since it cannot refuse c after doing a. Thus $ab+ac$ and $a(b+c)$ are not lighted-button experiment equivalent, and indeed the simple kind of experiment which distinguishes them established that they are not CSP equivalent.

Lighted-button experiments actually make more distinctions than CSP, but do not yet represent the full experimental scenario we wish to examine. For example, it is easy to check that the following holds.

Lemma 2.2. *The processes $abc+abd$ and $a(bc+bd)$ are equivalent with respect to all lighted-button experiments.*

Nevertheless, there is a simple experiment distinguishing these processes. We imagine equipping the experimenter with a *duplicator*, allowing him to copy the machine at any time, and perform independent experiments on the copies. Equivalently, we allow him to save and restore states of the machines. For example, we might think of implementing such experiments in software using an operating system fork.

The typical sort of *duplicator*-experiment looks something like the following.

(1) Press the a-button on P, and call the resulting machine P_a. Fail if the a-button cannot be pushed.
(2) Make two copies of P_a, call them P_{a1} and P_{a2}. This step cannot fail.
(3) Press the b-button and then the c-button of P_{a1}. Fail if either cannot be pushed.
(4) Press the b-button and then the c-button of P_{a2}. Fail if either the b-button cannot be pushed, or the c-button can be pushed.
(5) Succeed if none of the previous steps have failed.

The process $a(bc+cd)$ can pass this test, but the process $abc+abd$ cannot. So duplication increases the power of a lighted-button experimenter. In fact, equivalence with respect to duplicator experiments is precisely ready simulation; this is a corollary to Theorem 2.7.

It is well-known that understanding bisimulation in general seems to require exploring the behavior of all the children of a process. We present an excessively powerful form of duplicator experiment, called *wild duplicator experiments*, in which the experimenter is allowed to make any quantity (not necessarily finite) of copies of the process at each stage, and perform separate experiments on the copies. In particular, it is possible for the experimenter to see all the children of a process. We will show that this form of duplicator still only observes ready simulation.

We will allow infinite numbers of tests, and arbitrary Boolean combinations of the results. We do not restrict infinities to be countable. The infinities allow the results of this section to apply to arbitrarily branching processes, and *a fortiori* to finitely branching processes.

However, even wild duplicator experiments still do not explain bisimulation. It is easy to exhibit ready simulations between the nonbisimilar processes $a(bc+bd)$

and $abc + a(bc + bd) + abd$ of Fig. 1, and so by Theorem 1.3 they cannot be distinguished by duplicator experiments.

Let **B** be the set $\{\mathrm{tt}, \mathrm{ff}\}$ of Booleans; we use tt for success and ff for failure of experiments.

Definition 2.3. A *wild duplicator experiment* is an ordered tree, possibly countably deep and arbitrarily wide, with node labels and branching as follows. Each node v is labelled either choose, $a \in \mathsf{Act}$, or $B : \mathbf{B}^\kappa \to \mathbf{B}$, where κ is a cardinal ≥ 0, such that

(1) If v is labelled a, then v has exactly two children v_+ and v_-;
(2) If v is labelled $B : \mathbf{B}^\kappa \to \mathbf{B}$, then v has κ children, where κ is any ordinal ≥ 0 and if $\kappa = 0$ then v is a leaf node;
(3) If v is labelled choose then v may have any positive cardinality of children.

The intent is that nodes v labelled with actions a involve pushing the a-button on the process. If the button can be pushed, then the experimenter proceeds with v_+ on the resultant process; otherwise, the experimenter performs v_- on the unchanged original process. Nodes labelled with κ-ary Boolean functions instruct the experimenter to make κ copies of the process, perform the appropriate experiment on each copy, and combine the results by applying B. Nodes labelled choose allow the experimenter to choose one of the children and perform that experiment.

A simple duplication is modeled by a Boolean node. For example, consider an experiment which makes two copies of the process, runs test E_1 and E_2 on them, and succeeds iff both tests succeed; this is formalized by a node labelled by the binary **and** function, with children given by the formalizations of E_1 and E_2. The wild duplicator, which produces some unknown number of children, is a choice node with one child for the experiment to be performed on each number of children. For example, if the experiment is "Wild-duplicate P, and perform E on each copy, succeeding iff each copy succeeds", then the formalization starts with a single choose node with countably many children, the nth of which is an n-fold **or** node.

We write root(E) for the root node of the tree E.

Definition 2.4. We define $P \Uparrow E$ (resp. $P \Downarrow E$), pronounced "P can pass (resp. fail) E", iff there is some partial function ζ from the nodes of E to pairs of truth values

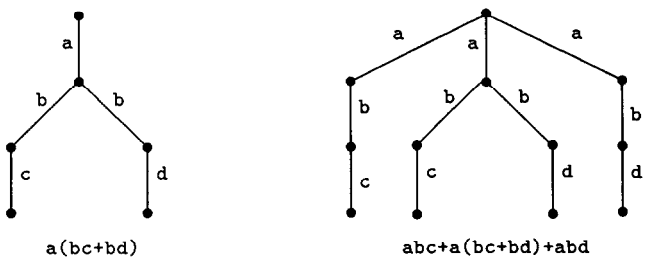

Fig. 1. Ready similar but not bisimilar.

and processes, such that $\zeta(\text{root}(E)) = \langle \text{tt}, P\rangle$, (resp. $\langle \text{ff}, P\rangle$) and whenever $\zeta(\nu) = \langle b, R\rangle$ we have the following.

- If ν is labelled a, then either
  - there is some R' such that $R \xrightarrow{a} R'$ and $\zeta(\nu_+) = \langle b, R'\rangle$; or
  - $R \xrightarrow{a}$, and $\zeta(\nu_-) = \langle b, R\rangle$.
- ν is labelled by $B : \mathbf{B}^\kappa \to \mathbf{B}$, and there is some κ-length vector $\boldsymbol{b}$ of Booleans such that $B(\boldsymbol{b}) = b$ and the αth child ν_α of ν has $\zeta(\nu_\alpha) = \langle b_\alpha, R\rangle$.
- ν is labelled choose, has κ children ν_α, and there is some $\beta < \kappa$ such that $\zeta(\nu_\beta) = \langle b, P\rangle$.

We say that ζ *demonstrates* that $P \Uparrow E$ (resp. $P \Downarrow E$).

Intuitively, ζ assigns to each node a process and the success or failure of the experiment given by that node on that process; the consistency conditions chart the progress of the experiment. It is possible that both $P \Uparrow E$ and $P \Downarrow E$; consider the wild duplicator experiment which succeeds precisely when the duplicator produces a prime number of copies. It is also possible to define experiments which can report both success and failure from the same sequence of actions of the process; e.g., the experiment which consists of infinitely often duplicating the process and never actually letting it perform an action.

We define $P \leqslant_{\text{Wild}} Q$ as for all experiments E, whenever $P \Uparrow E$ then $Q \Uparrow E$. For each experiment E there is an experiment $\neg E$ such that $P \Uparrow \neg E$ iff $P \Downarrow E$ and vice versa; take $\neg E$ to be the experiment E with an extra root node labelled by the negation function. Thus, we lose no generality by considering only successes.

We will need to construct demonstrations ζ; the following lemma makes the construction easier.

Definition 2.5. The function f is a *consistent choice function* for $P \sqsubseteq P'$ if f is a function from the descendants of P to those of P', such that $f(P) = P'$ and for all $Q \xrightarrow{a} R$ descendants of P, the following holds:

$$\begin{array}{ccc} Q & \sqsubseteq & f(Q) \\ \Big\downarrow{\scriptstyle a} & & \Big\downarrow{\scriptstyle a} \\ R & \sqsubseteq & f(R) \end{array}$$

Lemma 2.6. *Suppose that P and P' are arbitrarily-branching trees such that $P \sqsubseteq P'$. Then there is a consistent choice function for $P \sqsubseteq P'$.*

Proof. We may nonconstructively build such a function f by the Axiom of Choice. Let $\leqslant$ be a well-ordering of the descendants of P'. Define a sequence of partial functions f_i, taking the ith level of P (counting the root as level 1) to that of P', as follows. Let $f_1(P) = P'$. Suppose that R is on the nth level of P, and $R \xrightarrow{a} S$. Let $f_{n+1}(S)$ be S', the $\leqslant$-first descendant of $f_n(R)$ such that $S \sqsubseteq S'$; there is at least one

such S' by the fact that $R \sqsubseteq f(R)$. Let $f(T) = f_n(T)$ where T is on the nth level of P. It is easy to see that f is a consistent choice function for $P \sqsubseteq P'$. □

Theorem 2.7. *For all (arbitrarily branching) processes P and Q,* $P \leqslant_{\text{Wild}} Q$ *iff* $P \sqsubseteq Q$.

Proof. We first show that $\leqslant_{\text{Wild}}$ is a ready simulation relation, and hence $P \leqslant_{\text{Wild}} Q$ implies $P \sqsubseteq Q$. Suppose that $P \leqslant_{\text{Wild}} Q$ and $P \xrightarrow{a} P'$; we must show $Q \xrightarrow{a} Q'$ for some Q' such that $P' \leqslant_{\text{Wild}} Q'$. Suppose that there were no such Q'. Then for each a-child Q'_α, we have $P' \not\leqslant_{\text{Wild}} Q'_\alpha$; thus, there is some experiment E_α such that $P' \Uparrow E_\alpha$ but not $Q'_\alpha \Uparrow E_\alpha$. Let E' be the experiment which takes the conjunction of all the E_α. Then $P' \Uparrow E'$, but no Q_α can pass E'. Let E be the experiment which starts by pushing the α-button, and then running E'; P can pass E, but Q cannot. This violates the hypothesis that $P \leqslant_{\text{Wild}} Q$.

For the other clause of ready simulation, suppose that $P \leqslant_{\text{Wild}} Q$ and $P \xrightarrow{a}$. Then P can pass the experiment which pushes the a-button, failing if it can be pushed and succeeding if it cannot. Q must pass this experiment as well; hence $Q \xrightarrow{a}$.

For the converse, suppose that $P \sqsubseteq Q$ and $P \Uparrow E$. Let ζ be any function demonstrating that $P \Uparrow E$. We will construct a function ζ' demonstrating that $Q \Uparrow E$. Let f be a consistent choice function for $P \sqsubseteq Q$. Define

$$\zeta'(\nu) = \begin{cases} \langle b, f(R)\rangle & \text{if } \zeta(\nu) = \langle b, R\rangle \text{ and } R \in \text{descendants}(P), \\ \langle b, R\rangle & \text{if } \zeta(\nu) = \langle b, R\rangle \text{ and } R \notin \text{descendents}(P), \\ \text{undefined} & \text{otherwise.} \end{cases}$$

It is straightforward to check that ζ' demonstrates that $Q \Uparrow E$. □

It is worth noting that simple duplicator experiments (with binary Boolean operations and no choice nodes) suffice to capture ready simulation of finitely-branching processes; see [6] for more details.

3. Global testing experiments

For any process P, let $\text{Succ}_a(P) = \{P' : P \xrightarrow{a} P'\}$. In our setting, this set of successor processes of P will always be finite.[3] Notice that a duplicator, by making $|\text{Succ}_\alpha(P)|$ copies of P and pressing an a-button on each copy, has the possibility of getting the entire set $\text{Succ}_a(P)$ to experiment upon. Milner's experimental explanation of bisimulation reveals that the experimenter must, however, do more than merely have the *possibility* to see all the successors—he must *know* when he has seen them all.

[3] Also, as we are taking synchronization trees as our basic semantics, there may be several isomorphic elements of $\text{Succ}_a(P)$. In particular, the process $a + a$ has a synchronization tree with two leaves, which are isomorphic but not equal; $\text{Succ}_a(a + a)$ is the (two-element) set containing those two leaves.

This is formalized in [18], where Milner describes a mechanism for exploring all the alternatives available from a given state, by allowing variation of some "ambient ('weather') conditions" which determine which nondeterministic choice the machine will take. We formulate 'weathers' in terms of a *global-testing duplicator* [1]. The global-testing duplicator is a device with a chamber, a control panel with one button per action, and a chute. The experimenter places the machine in the chamber, and presses a button on the control panel, say the a-button. Out of the chute drops one copy of each a-descendant of the process.

However, global-testing goes too far; global-testing equivalence is strictly finer than bisimulation. The experimenter can simply count the black boxes that come out of the chute. In Milner's metaphor, this is counting the number of varieties of weather available to investigate. In fact, we have the following theorem.

Theorem 3.1. *Global-testing duplicator experiment equivalence coincides with isomorphism of unordered trees. In particular, the bisimilar processes a and $a+a$ are not global-testing duplicator equivalent.*

The test which distinguishes them is: "put the process in the global-testing duplicator. Press the a-button. Succeed iff one box comes out the chute".

Actually this simple form of global-testing duplicator does not precisely match Milner's weather scenario: two different forms of weather may drive the process into the same state. Milner's description directly corresponds to the *wild global-testing duplicator*, which may produce *one or more* copies of each descendent of its input. This uncertainty blurs the counting of successor processes, raising the *prima facie* possibility that nonisomorphic process trees might be identified.

The same experiment distinguishes a from $a+a$, although in a slightly different way. Now, a may pass the experiment, although it will no longer pass it in every run; however $a+a$ must fail in every run.

In fact, the wildness does not blur any distinctions at all, as is seen in the following result.

Theorem 3.2. *Wild global-testing duplicator equivalence coincides with unordered tree isomorphism.*

We will first define a partial order $P \preceq Q$ on finite trees. Let $P \upharpoonright n$ be P truncated at depth n. Unlike most comparisons between processes we have considered, $\preceq$ is antisymmetric; $P \preceq Q$ and $Q \preceq P$ will imply $P \equiv Q$ (viz. that P and Q are isomorphic synchronization trees). We will construct experiments $E_{Q,n}$ such that P can pass $E_{Q,n}$ iff $P \upharpoonright n \preceq Q \upharpoonright n$. If P and Q are distinct synchronization trees, then for some n we have $P \upharpoonright n \not\equiv Q \upharpoonright n$, and so either $P \upharpoonright n \not\preceq Q \upharpoonright n$ or $Q \upharpoonright n \not\preceq P \upharpoonright n$. So, $E_{Q,n}$ or $E_{P,n}$ will distinguish P and Q, and the theorem will follow. The necessary mathematics will take up the rest of this section.

First, our partial order on finite trees. The condition $P \preceq Q$ holds if, informally, Q can be obtained by repeated duplication of subtrees of P. Formally, this is defined by induction on the depth of finite trees.

Definition 3.3. $\mathbf{0} \preceq \mathbf{0}$, and whenever

$$P = \sum_{a \in \mathrm{Act}} \sum_{i=1}^{p_a} aP_{ai}, \qquad Q = \sum_{a \in \mathrm{Act}} \sum_{i=1}^{q_a} aQ_{aj} \tag{*}$$

are such that

(1) for each Q_{aj} there is some P_{ai} such that $P_{ai} \preceq Q_{aj}$,

(2) there is a 1-1 function $f:[1 \ldots p_a] \to [1 \ldots q_a]$ such that $P_{aj} \preceq Q_{a,f(j)}$,

then $P \preceq Q$.

That is, each child of Q has a "cousin" which is a child of P, and distinct children of P have distinct cousins in Q. It is easy to show that $\preceq$ is a preorder.

Lemma 3.4. *$\preceq$ is a partial order, and a congruence with respect to $a(\cdot)$, $+$, and $\restriction n$.*

Proof. Reflexitivity, transitivity, and congruence are straightforward. By a predictable induction on n, we show that it is antisymmetric; that is, if $P \preceq Q \preceq P$ then P and Q are isomorphic synchronization trees. This is trivial if $P = Q = \mathbf{0}$. Let P and Q be expressed as in (*), fix an action a and let $f:[1 \ldots p_a] \to [1 \ldots q_a]$ and $g:[1 \ldots q_a] \to [1 \ldots p_a]$ be the 1-1 functions showing that $P \preceq Q$ and $Q \preceq P$ respectively. The existence of 1-1 functions shows that $p_a \leq q_a \leq p_a$ and hence $p_a = q_a$. Therefore f and g are also onto, and so fg and gf are permutations. Recall that if h is a permutation of a finite set and i is an element of that set, then the set $\{i, h(i), h^{(2)}(i), \ldots\}$ is a finite set, called the *orbit* of i under h; as h is 1-1, we must have $h^{(k)}(i) = i$ for some k, called the *period* of i.

Fix i. We have

$$P_{ai} \preceq Q_{a,f(i)} \preceq P_{a,gf(i)} \preceq \cdots \preceq P_{a,(gf)^{(k)}(i)} = P_{a,i}$$

where k is the period of i. By transitivity, we have $Q_{a,f(i)} \preceq P_{a,i}$, and so by induction $P_{a,i} \equiv Q_{a,f(i)}$. With a little bit of work, this establishes a bijection between the children of P and those of Q as desired. □

We now define the experiments $E_{Q,n}$ such that P can pass $E_{Q,n}$ iff $P \restriction n \preceq Q \restriction n$. The experiment $E_{Q,0}$ always succeeds. To see if P can pass $E_{Q,n+1}$, wild-duplicate P under each action a, giving $P'_{a1}, \ldots, P'_{a,p'_a}$. If $p'_a \neq q_a$ then the experiment fails. If $p'_a = q_a$ then for each i, see if P'_{ai} passes $E_{Q_{ai},n}$. $E_{Q,n+1}$ succeeds if each $E_{Q_{ai},n}$ succeeds.

Lemma 3.5. *P can pass $E_{Q,n}$ iff $P \restriction n \leq Q \restriction n$.*

Proof. This clearly is true for $n = 0$. For greater n, suppose that P can pass $E_{Q,n}$. Let P and Q be given as in (*). Suppose that the wild global-testing duplicator produced $\langle P'_{aj} : a \in \mathsf{Act}, j \in [1 \ldots q_a] \rangle$, where each child of P appears in this listing at least once. We have $P \leq \sum_{a,j} aP'_{aj}$. As each P'_{aj} passes $E_{Q_{aj},n-1}$ we have $P'_{aj} \restriction (n-1) \leq Q_{aj} \restriction (n-1)$ by induction, and hence we have

$$P \restriction n \leq \sum_{a,j} aP'_{aj} \restriction (n-1) \leq \sum_{a,j} aQ_{aj} \restriction (n-1) = Q \restriction n$$

as desired. The other direction is similar. □

Lemma 3.6. *P and Q and agree on all wild global-testing duplicator experiments iff $P \equiv Q$.*

Proof. Clearly the result of performing an experiment on a process depends only on its synchronization tree. Conversely, suppose that $P \not\equiv Q$. Then there is some n such that $P \restriction n \not\equiv Q \restriction n$. By antisymmetry, we have $P \restriction n \not\leq Q \restriction n$ or $Q \restriction n \not\leq P \restriction n$; suppose the former. Then by Lemma 3.5, we know that P cannot pass $E_{Q,n}$ but Q can. Hence P and Q are distinguishable with a wild global-testing experiment. □

4. Modal logic

Modal logics which arise naturally in process specification are intimately connected to experimental equivalence.

It is possible to give a straightforward logic for ready simulation, which we call *denial logic* (called "limited modal logic" in [7]). Disjunction does not increase the descriptive power of denial logic, because the law $\langle a \rangle(\varphi \vee \psi) = (\langle a \rangle \varphi) \vee (\langle a \rangle \psi)$ is valid. The syntax of *denial logic* is

$$\varphi \ ::= \ \text{Can't}(a) \mid \mathrm{tt} \mid \varphi \wedge \varphi \mid \langle a \rangle \varphi$$

and satisfaction is defined as usual:

- $P \models \mathrm{tt}$ always,
- $P \models \varphi \wedge \psi$ iff $P \models \varphi$ and $P \models \psi$,
- $P \models \langle a \rangle \varphi$ iff for some P', $P \xrightarrow{a} P'$ and $P' \models \varphi$,
- $P \models \text{Can't}(a)$ if $P \not\xrightarrow{a}$.

Formulas of denial logic correspond quite naturally to a certain set of "elementary" duplicator experiments. For example, Can't(a) is an experiment which succeeds if the a button cannot be pressed, and $\varphi \wedge \psi$ is an experiment which starts by duplicating the process, and then performing appropriate experiments on the copies. In this way we can show that equivalence with respect to denial logic is identical to equivalence with respect to duplicator experiments.

An even more natural *Hennessy-Milner logic* (*HML*) logic is well known to yield to bisimulation [13]. The syntax of HML is

$$\varphi ::= \mathrm{tt} \mid \mathrm{ff} \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \langle a \rangle \varphi \mid [a]\varphi.$$

The significant new clause in the definition of satisfaction is $P \models [a]\varphi$ iff for all P' such that $P \xrightarrow{a} P'$, $P' \models \varphi$. Note that the denial formula Can't(a) is expressible in HML as $[a]\mathrm{ff}$, so denial formulas can be seen as a special case of HML formulas.

Now it is straightforward to seen how to distinguish non-bisimilar processes using a (modal) global-testing duplicator. Processes P and Q are not bisimilar iff there is some Hennessy-Milner formula φ which distinguishes them, say $P \models \varphi$ and $Q \not\models \varphi$. We can construct an experiment e_φ from φ on which P can succeed, but Q never will. For example, the experiment $e_{\langle a \rangle \varphi}$ starts with pressing the a-button, and then performs e_φ on the resulting machine (or failing if the a-button cannot be pressed). Dually, the experiment $e_{[a]\varphi}$ does an a-button global-test duplication of the machine in the chamber and checks to see that every machine coming out of the chute passes e_φ.

5. Conclusion

We have extended the notion of experimental equivalence of processes by considering experiments in which processes can be duplicated, allowing collection of information about alternative process behaviors during an experiment. This idea was originally suggested by Milner as yielding an experimental understanding of bisimulation between processes. The authors have found it hard to provide a physical justification or operational rationale for a key technical restriction used in Milner's experimental characterization of bisimulation—that experimental outcomes only be combined "modally". The apparently more natural variations of duplicating-experiment equivalence we have examined—ready similarity and unordered tree isomorphism—respectively strictly coarsen and strictly refine bisimulation.

Bisimulation has been singled out by Milner as the finest notion of equivalence of synchronization trees which seems suitable for interleaving process theory. It might help clarify the theoretical role of bisimulation if this observation was rigorously formulated, but the authors do not question that bisimulation arises as a fundamental concept of interleaving concurrency theory. This is especially clear in the modal characterization of bisimulation via Hennessy-Milner logic, which is more elegant than the corresponding characterization of ready similarity.

The debate about the proper choice of process equivalence continues to be technically fruitful. Recently, Larsen and Skou [16] have argued that bisimulation can be understood experimentally in the setting of experiments on *probabilistic* processes (cf. [8]), and Vaandrager and Groote [11] propose other equivalences finer than ready simulation, but still coarser than bisimulation, based on a relaxed SOS discipline.

The authors interpret these results—even Larsen and Skou's (although they do not agree)—as indicating that bisimulation has not been persuasively justified as a *computational* equivalence based on effectively observing and experimenting with processes. Consequently, while making full use of the richly developed bisimulation based methods which Milner's school has shown are widely applicable, researchers and protocol specifiers and verifiers should anticipate limitations of bisimulations. Protocols may be expected to be correct in senses weaker than bisimulation, so when bisimulation fails to hold, one should not hesitate to try establishing correctness in terms of suitable experiment-based equivalences.

We are continuing to investigate the theory of ready simulation. For example, we have a finite axiomatization of ready simulation on finite trees and a naive polynomial-time algorithm for deciding ready simulations of finite-state processes. A major, crucial, development yet to be undertaken is the extension of the theory of ready simulation to handle hidden moves.

6. Acknowledgement

Robin Milner has been immensely helpful to this study, and quite generous in giving clarifications of [18]. We are grateful to Kim Larsen, Sorin Istrail and Miller Maley for several discussions. Finally, we would like to thank Marta Kwiatkowska and an anonymous referee for much constructive criticism.

References

[1] S. Abramsky, Observation equivalence as a testing equivalence, *Theoret. Comput. Sci.* **53** (1987) 225-241.

[2] S. Abramsky and S. Vickers, Quantales, observational logic and process semantics, Research Report DOC 90/1, Imperial College, London, 1990.

[3] D. Austry and G. Boudol, Algèbre de processus et synchronisation, *Theoret. Comput. Sci.* **30** (1984) 91-131.

[4] J.C.M. Baeten and R.J. van Glabbeek, Another look at abstraction in process algebra, in: T. Ottman, ed., *14th ICALP*, Lecture Notes in Computer Science, Vol. 267 (Springer, Berlin, 1987) 84-94.

[5] J.A. Bergstra and J.W. Klop, Process algebra for synchronous communication, *Inform. and Control*, **60** (1984) 109-137.

[6] B. Bloom, Ready simulation, bisimulation, and the semantics of CCS-like languages, Ph.D. thesis, Massachusetts Institute of Technology, August 1989.

[7] B. Bloom, S. Istrail and A.R. Meyer, Bisimulation can't be traced: Preliminary report, in: *Proc. 15th Symp. Principles of Programming Languages* (ACM, 1988) 229-239; final version in preparation for journal submission.

[8] B. Bloom and A.R. Meyer, A remark on the bisimulation of probabilistic process, in: *A. Meyer and M. Taitslin*, eds, *Proc. Logic at Botik '89*, Lecture Notes in Computer Science, Vol. 363 (Springer, Berlin, 1989) 26-40.

[9] R. de Simone, Higher-level synchronising devices in MEIJE-SCCS, *Theoret. Comput. Sci.* **37** (1985) 245-267.

[10] R. DeNicola and M.C. Hennessy, Testing equivalences for processes, *Theoret. Comput. Sci.* **34** (1984) 83-133.

[11] J.F. Groote and F. Vaandrager, Structured operational semantics and bisimulation as a congruence (extended abstract), in: G. Ausiello, M. Dezani-Ciancaglini and S. Ronchi della Rocca, eds., *Proc. 16th Internat. Coll., Automata, Languages and Programming*, Lecture Notes in Computer Science, Vol. 372 (Springer, Berlin, 1989) 626–638.

[12] M. Hennessy, *Algebraic Theory of Processes* (MIT Press, Cambridge, MA, 1988).

[13] M.C.B. Hennessy and R. Milner, On observing nondeterminism and concurrency, in: J. de Bakker and J. van Leeuwen, eds., *Proc. Internat. Coll. Automata, Languages and Programming*, Lecture Notes in Computer Science, Vol. 85 (Springer, Berlin, 1980) 299–309.

[14] C. Hoare, Communicating sequential processes, *Comm. ACM.* **21** (1978) 666–677.

[15] C.A.R. Hoare, *Communicating Sequential Processes* (Prentice-Hall, Englewood Cliffs, NJ, 1985).

[16] K. Larsen and A. Skou, Bisimulation through probabilistic testing (preliminary report), in: *Proc. 16th Symp. Principles of Programming Languages* (ACM, 1989) 344–352.

[17] R. Milner, *A Calculus of Communicating Systems*, Lecture Notes in Computer Science, Vol. 92 (Springer, Berlin, 1980).

[18] R. Milner, A modal characterisation of observable machine-behavior, in: E. Astesiano and C. Böhn, eds., *Proc. CAAP '81*, Lecture Notes in Computer Science, Vol. 112 (Springer, Berlin, 1987) 25–34.

[19] R. Milner, Calculi for synchrony and asynchrony, *Theoret. Comput. Sci.* **25** (1983) 267–310.

[20] R. Milner, Private communication, 1991.

[21] I. Phillips, Refusal testing, in: L. Knott, ed., *Proc. 13th ICALP*, Lecture Notes in Computer Science, Vol. 226 (Springer, Berlin, 1986) 304–313.

[22] I. Phillips, Refusal testing, *Theoret. Comput. Sci.* **50** (1987) 241–284.

[23] G.D. Plotkin, A structural approach to operational semantics, Technical Report DAIMI FN-19, Computer Science Dept., Aarhus Univ., 1981.

[24] A. Pnueli, Linear and branching structures in the semantics and logics of reactive systems, in: W. Brauer, ed. *Proc. Internat. Coll. Automata, Languages and Programming*, Lecture Notes in Computer Science, Vol. 194 (Springer, Berlin, 1985) 15–32.