Math. Struct. in Comp. Science (2003), *vol.* 13, *pp.* 619–656. © 2003 Cambridge University Press
DOI: 10.1017/S0960129503004018 Printed in the United Kingdom

The expressive power of urgent, lazy and busy-waiting actions in timed processes[†]

FLAVIO CORRADINI and DINO DI COLA

Dipartimento di Informatica, Università degli Studi di L'Aquila, Via Vetoio, Loc. Coppito, L'Aquila, Italy
Email: {flavio,dicola}@edi.univaq.it

Received 6 January 2001; revised 10 June 2002

In this paper we show how the expressive power of a language for the description of timed processes strongly affects the discriminating power of urgent and patient actions. In a sense, it studies the interplay between syntax and semantics of time-critical systems.

1. Introduction

In recent years several well-known formalisms that are suitable for the specification and verification of concurrent systems (logics, process algebras, Petri nets, *etc.*) have been extended to cope with time-critical systems. The correctness of time-critical systems depends not only on *which* actions these systems can perform but also on *when* such actions are performed. The problem is that they may enter an incorrect state if a particular action is performed too early or too late.

In this paper we present a study of the relationships between the syntax and semantics of time-critical systems. In more detail, we show how the expressive power of the language for the description of time-critical systems strongly affects their timing/performance aspects.

The (*CCS*-like) language we consider is quite expressive. It has durational actions as in Aceto and Murphy (1996) and Gorrieri *et al.* (1995), and facilities for delaying processes as in Hennessy and Regan (1995), Moller and Tofts (1990) and Yi (1990) (still focusing on timed *CCS*-like languages). Processes are compared according to performance congruence. This equivalence was introduced in Corradini (1998) and discriminates processes according to their *functionality* (which actions they can perform) and *performance* (a measure of the time consumed during their execution). Three kinds of basic actions naturally reside within the above-mentioned framework: eager actions (those that are performed as soon as they can – these are also called urgent actions), lazy actions (those that can be delayed before their execution) and busy-waiting actions (those which denote synchronisations between two system components). These kinds of action emerge as classes of tests (experiments) to exercise our processes in order to decide on their equivalence (performance/time-sensitivity, in the present setting).

[†] Research supported by Murst progetto 'Saladin: Software Architectures and Languages to Coordinate Distributed Mobile Components'.

https://doi.org/10.1017/S0960129503004018 Published online by Cambridge University Press

We show how the discriminating power of urgent, lazy and busy-waiting actions changes depending on the expressiveness of the language for the description of time-critical systems. This study is conducted by showing how three bisimulation-based equivalences relate when the language changes according to four significant features. The three equivalences are[†]:

— *performance congruence* (Corradini 1998), which is obtained by carrying out eager, lazy and busy-waiting tests,
— *eager equivalence* (Gorrieri *et al.* 1995), which is obtained by carrying out eager and busy-waiting tests, and
— *lazy equivalence*, which is obtained by carrying out lazy and busy-waiting tests.

The four language features have to do with non-deterministic composition, relabelling functions (Milner 1989), the number of actions a process can perform at a given time and the nature of actions (visible/invisible). We show how the three equivalences are related when:

— The language allows choices at the same time or, also, at different times. In other words, we are distinguishing between 'timed alternative compositions' and 'alternative timed compositions'. In the former case, the non-deterministic composition only involves the functionality of the process, while in the latter it involves both functionality and timing. For example, I can choose at time t between a snack and a full lunch $((\text{snack} + \text{lunch})@t)$ or I can choose between a snack at noon and a dinner eight hours after $((\text{snack}@t) + (\text{dinner}@t')$, where $t' > t$).
— The language allows relabelling functions that preserve the duration of the actions (that is, they rename actions having the same duration) or, also, rename actions with (possibly) different durations.
— The language allows the description of processes that can perform finitely many actions (though of unbounded number) at a fixed time or, also, infinitely many[‡].
— The language allows only visible actions or, also, internal ones (such as synchronisations).

Note that these different languages do not constitute a hierarchy but a classification of specific language features that are significant when comparing the discriminating power of the urgent and patient actions.

It turns out that if the language allows:

(a) only visible actions,
(b) only processes that can perform finitely many actions at a fixed time,
(c) only choices at the same time, and
(d) only duration preserving relabelling functions,

then eager tests and lazy tests have the same discriminating power. In such languages, performance congruence, eager equivalence and lazy equivalence coincide.

[†] We do not consider equivalences without busy-waiting tests since they would lead to unwanted identifications.
[‡] This permits the construction of processes that can do infinitely many actions in a finite interval of time – these are also called *Zeno*-machines.

https://doi.org/10.1017/S0960129503004018 Published online by Cambridge University Press

If the language allows invisible actions and (b), (c), and (d) as above, then lazy tests are more discriminating than eager ones or, in other words, discriminating enough to capture the same equalities of performance congruence. As a consequence, in the definition of performance congruence, the lazy tests are the significant ones while the eager tests are superfluous.

If the language allows the description of processes that can perform infinitely many actions at a fixed time, choices at different times or relabelling functions that do not preserve the duration of the actions, then we only have the obvious implications; namely, performance congruence implies the two other equivalences.

It has to be noted that if eager equivalence and/or lazy equivalence do not coincide with performance congruence (and the language permits process synchronisation), then eager equivalence and lazy equivalence are not even congruences (they are not preserved by parallel composition with synchronisation), while performance congruence remains a congruence.

We now discuss our main motivations for this work:

— From a foundational point of view, it provides some insight into the relationships between syntax and semantics in a time-critical setting by showing how the discriminating power of timed actions may change according to the features of the base language. We explain, in a sense, how the semantics of time-critical systems change when their syntax changes. This should help in the design of new languages for the description of time-critical systems. The present work says, indeed, which semantics should (and should not) be coupled with the language being designed.
— It provides more confidence in the definition of a new performance-sensitive equivalence, namely performance congruence. It permits the removal of redundancy in its definition (by fixing the class of tests over processes strictly needed to decide their equivalence). For instance, if lazy tests together with eager tests are not more discriminating than just lazy tests, we could simply exercise our systems over the latter ones to deduce the same equivalences.
— In addition, it studies the relationships between three important timed equivalences. Recently (Bérard *et al.* 2000), the equivalence that we have called eager equivalence has been proved to be decidable in polynomial time over a process algebra with only visible durational actions. Using the same process algebra, we prove that eager equivalence, lazy equivalence and performance congruence coincide. Hence, we also have a polynomial time algorithm for deciding both lazy equivalence and performance congruence.

The rest of the paper is organised as follows. The next section contains the required background for our study, and Section 3, the core of the paper, relates the three equivalences over the different languages. Concluding remarks are given in Section 4. Detailed proofs of two main results (Theorem 3.1 and Theorem 3.2) are given in appendices, allowing you to skip their technical development if you wish. Note, however, that some intermediate results are interesting in their own right (for example, the decomposition result for parallel timed systems stated in Lemma B.7).

https://doi.org/10.1017/S0960129503004018 Published online by Cambridge University Press

2. A theory of timed processes

The process algebra we consider follows the same descriptive style as those in Aceto and Murphy (1996), Ferrari and Montanari (1995) and Gorrieri *et al.* (1995). We now briefly discuss its main assumptions and contrast them with those behind a different approach to the specification of timed system; namely, the *two-phase* functioning principle (see Hennessy and Regan (1995), Moller and Tofts (1990) and Yi (1990), to cite a few timed *CCS*-like languages). This should clarify our design decisions (see Corradini (2000) for a more comprehensive comparison).

Actions are durational. Basic actions take time to be performed (that is, actions have a duration) and time passes in a system only due to the execution of these actions. This contrasts with the approach taken in Hennessy and Regan (1995), Moller and Tofts (1990) and Yi (1990), where basic actions are instantaneous events. The passage of time is explicitly modelled by a special 'tic' action upon which all components of a global system must synchronise.

Time is absolute. During a system execution, time stamps are associated with the observed events. These time stamps are referred to the starting time of the system execution; that is, time features as *absolute time*. In Hennessy and Regan (1995), Moller and Tofts (1990) and Yi (1990), the (passage of) time refers to the execution time of the previous observed action. In this case time features as *relative time*.

Functional behaviour is timed. The description of the functional behaviour (that is, which actions a system can perform) and the description of time and time passing are given by integrating these two different aspects into a single framework. This framework takes the form of (a single) labelled transition system. In Hennessy and Regan (1995), Moller and Tofts (1990) and Yi (1990), functional behaviour and temporal behaviour are kept separate, using two orthogonal observation mechanisms (two labelled transition systems: one describing the functional bahaviour and the other describing the time and time passing). This is known in the literature as the *two-phase* functioning principle: a phase in which a system asynchronously evolves via execution of actions is followed by a phase in which conceptually time progresses.

Local clocks. A local clock is associated with each of the parallel components of a global system. These local clocks elapse independently of each other, although they define a unique notion of global time. In Hennessy and Regan (1995), Moller and Tofts (1990) and Yi (1990), the notions of global time and unique global clock are made explicit.

We also consider several extensions to the theory of timed processes in Aceto and Murphy (1996), Ferrari and Montanari (1995) and Gorrieri *et al.* (1995). First of all we allow process delay. Delays are not permitted in Aceto and Murphy (1996), Ferrari and Montanari (1995) and Gorrieri *et al.* (1995), though they are the most significant operation for modelling time and time passing within the category of process algebras with durationless actions, relative time, global clock, functional and timed behaviours described in two separate semantic framework (Hennessy and Regan 1995; Moller and

https://doi.org/10.1017/S0960129503004018 Published online by Cambridge University Press

Tofts 1990; Yi 1990). As expected, indeed, the addition of delay operators to our language causes very interesting effects for our study. Also, infinite versions of non-deterministic composition and parallel composition are of interest.

In the rest of this section we give the formal description of our *CCS*-like process description language, its operational semantics and performance congruence.

— We assume a set of actions A (ranged over by $\alpha, \beta, \ldots$) from which we obtain the set of co-actions $\bar{A} = \{\bar{\alpha} \mid \alpha \in A\}$. Act (ranged over by $a, b, \ldots$) stands for $A \cup \bar{A}$ and denotes the set of *visible* actions with the convention that $a \in Act$ implies $\bar{\bar{a}} = a$.
— The *invisible* action is denoted by $\tau \notin Act$. We use Act_τ (ranged over by $\mu, \mu', \ldots$) as the set of all actions $Act \cup \{\tau\}$.
— $\mathbb{N}$ and $\mathbb{N}^+$ (ranged over by $n, n', \ldots$), respectively, denote the set of natural numbers and the set of positive ones.
— *Durational functions* (ranged over by $f, g, \ldots$) associate to each action the time units needed for its execution. In the rest of the paper a durational function $f : Act \to \mathbb{N}^+$ is chosen simply to fix this parameter. We assume $f(a) = f(\bar{a})$, for each $a \in Act$.
— For each $n \in \mathbb{N}^+$, $Act(n) = \{a \in Act \mid f(a) = n\}$ defines the set of all visible actions with duration n.
— *Relabelling functions* (ranged over by $\Phi, \Phi', \ldots$) are used to rename actions in Act_τ. We assume a relabelling function $\Phi : Act_\tau \to Act_\tau$ with the convention that $\Phi(\tau) = \tau$ and $\overline{\Phi(a)} = \Phi(\bar{a})$ for each $a \in Act$.
— Var denotes the set of process *variables* (ranged over by $x, y, \ldots$) used for recursive definitions.

Let $\mathcal{Q}$ (ranged over by $q, q', \ldots$) denote the set of terms generated by the following grammar:

$$q ::= nil \mid a.q \mid (n)q \mid \sum_{i \in I} q_i \mid \prod_{i \in I} q_i \mid q \setminus B \mid q[\Phi] \mid x \mid \text{rec } x.q$$

where $n \in \mathbb{N}^+$, $I \subseteq \mathbb{N}$, $B \cup \{a\} \subseteq Act$, $x \in Var$ and Φ is a relabelling function. We assume the usual notions of free variables and bound variables in a term. Given $q \in \mathcal{Q}$, $\mathcal{F}(q)$ denotes its set of free variables. The set of closed $\mathcal{Q}$ terms, also called *processes*, is denoted by $\mathcal{P}$. In the rest of the paper we concentrate on $\mathcal{P}$ terms (ranged over by $p, p', \ldots$) unless we specify otherwise.

Process *nil* denotes a terminated process. By prefixing a process p with an action a, we get a process $a.p$, which performs an action a and then behaves like p. $(n)p$ denotes a process that delays the execution of p by n time units. $\sum_{i \in I} p_i$ denotes the alternative composition of p_i, namely term $p_1 + \ldots + p_i + \ldots$ where $i \in I$. $\prod_{i \in I} p_i$ denotes the parallel composition of p_i, namely term $p_1 | \ldots | p_i | \ldots$ where $i \in I$. In both cases, we require $|I| \geqslant 2$. $p \setminus B$ is a process that behaves like p, except that actions in B, and their complements, are not allowed. $p[\Phi]$ behaves like p but its actions are relabelled according to relabelling function Φ. Finally, rec $x.p$ is used for recursive definitions. For the sake of simplicity, terminal *nil*s can be omitted; for example, $a + b.c$ stands for $a.nil + b.c.nil$.

https://doi.org/10.1017/S0960129503004018 Published online by Cambridge University Press

Table 1. *Clock Distribution Equations.*

$$\begin{array}{rcl} t + n \Rightarrow p & = & t \Rightarrow (n)p \\ t \Rightarrow (p_1 \mid p_2) & = & (t \Rightarrow p_1) \mid (t \Rightarrow p_2) \\ t \Rightarrow (p_1 + p_2) & = & (t \Rightarrow p_1) + (t \Rightarrow p_2) \\ t \Rightarrow (p \setminus B) & = & (t \Rightarrow p) \setminus B \\ t \Rightarrow (p[\Phi]) & = & (t \Rightarrow p)[\Phi] \end{array}$$

2.1. *The operational semantics*

$\mathscr{P}$ is equipped with an *SOS* semantics in terms of labelled transition systems. The states of these systems are terms of a syntax extending that of processes with a local *clock prefixing* operator ($t \Rightarrow \_$) that records the evolution of the various parts of a distributed state. More precisely, the set of *states* (denoted by $\mathscr{D}$ and ranged over by $d_1, d_2, \ldots$) contains terms generated by the following syntax:

$$d ::= t \Rightarrow nil \;\Big|\; t \Rightarrow a.p \;\Big|\; t \Rightarrow (n)p \;\Big|\; t \Rightarrow \mathrm{rec}\ x.p \;\Big|\; \sum_{i \in I} d_i \;\Big|\; \prod_{i \in I} d_i \;\Big|\; d \setminus B \;\Big|\; d[\Phi]$$

where $(n)p, \mathrm{rec}\ x.p \in \mathscr{P}$, $t \in \mathbb{N}$ and $B \cup \{a\} \subseteq Act$.

In order to define a simple transition relation, the shorthand expression $t \Rightarrow p$ is used to mean that t distributes over the operators, until the sequential components. The equations in Table 1, called *clock distribution equations*, show that a term $t \Rightarrow p$ can be reduced to a canonical state when we interpret these equations as rewrite rules from left to right.

The set of *labels* for the transition relation is $Act_\tau \times \mathbb{N} \times \mathbb{N}$. Each transition is of the form $d \xrightarrow{\langle \mu, t, r \rangle} d'$ with the intuitive meaning that state d can become state d' by performing an action μ at completion time t. r is the execution delay meaning that the execution of action μ is started r time units after the last performed action by the sub-process responsible for the execution of μ. The transition relation $d \xrightarrow{\langle \mu, t, r \rangle} d'$ is defined by the axioms and inference rules given in Table 2. It is worthwhile observing that these rules are parameterised on the chosen durational function f. Hence, we should write $\rightarrow_f$, but for the sake of simplicity, the subscript will always be omitted whenever it is clear from the context.

A few comments on the rules in Table 2 are now in order. The rule for action prefixing *Act* states that process $a.p$ with local clock t can complete the execution of action a at any time $t + f(a) + r$, where $r \geqslant 0$ is the delay before the execution of action a is started. Note that it may be that $(t + f(a) + r) \Rightarrow p$ is not a state; in such a case, applications of the clock distribution equations will eventually transform $(t + f(a) + r) \Rightarrow p$ into a state. Rule *Del* states that process $(n)p$ with local clock t can complete the execution of action μ at time t' and delay $r \geqslant 0$ if process p with local clock $t + n$ can do the same. In the premise of this rule, the term $n + t \Rightarrow p$ may need applications of the clock distribution equations to become a state. Rules Sum_1, Sum_2, *Rec*, Par_1, Par_2, *Res* and *Rel* for alternative composition, recursion, asynchronous execution of a parallel composition, restriction and relabelling are as usual. Only note that in the premise of rule *Rec*, the

https://doi.org/10.1017/S0960129503004018 Published online by Cambridge University Press

Table 2. *The Structural Rules for the Operational Semantics.*

$$Act \quad \frac{r \geqslant 0}{t \Rightarrow a.p \xrightarrow{\langle a,t+f(a)+r,r\rangle} (t+f(a)+r) \Rightarrow p}$$

$$Del \quad \frac{t+n \Rightarrow p \xrightarrow{\langle \mu,t',r\rangle} d}{t \Rightarrow (n)p \xrightarrow{\langle \mu,t',r\rangle} d}$$

$$Sum_1 \quad \frac{d_1 \xrightarrow{\langle \mu,t,r\rangle} d_1'}{d_1+d_2 \xrightarrow{\langle \mu,t,r\rangle} d_1'} \qquad Sum_2 \quad \frac{d_2 \xrightarrow{\langle \mu,t,r\rangle} d_2'}{d_1+d_2 \xrightarrow{\langle \mu,t,r\rangle} d_2'}$$

$$Rec \quad \frac{t \Rightarrow p[\text{rec } x.p/x] \xrightarrow{\langle \mu,t',r\rangle} d}{t \Rightarrow \text{rec } x.p \xrightarrow{\langle \mu,t',r\rangle} d}$$

$$Par_1 \quad \frac{d_1 \xrightarrow{\langle \mu,t,r\rangle} d_1'}{d_1 \mid d_2 \xrightarrow{\langle \mu,t,r\rangle} d_1' \mid d_2} \qquad Par_2 \quad \frac{d_2 \xrightarrow{\langle \mu,t,r\rangle} d_2'}{d_1 \mid d_2 \xrightarrow{\langle \mu,t,r\rangle} d_1 \mid d_2'}$$

$$Synch \quad \frac{d_1 \xrightarrow{\langle a,t,r_1\rangle} d_1',\ d_2 \xrightarrow{\langle \bar{a},t,r_2\rangle} d_2',\ (r_1=0 \text{ or } r_2=0)}{d_1 \mid d_2 \xrightarrow{\langle \tau,t,0\rangle} d_1' \mid d_2'}$$

$$Res \quad \frac{d \xrightarrow{\langle \mu,t,r\rangle} d' \quad \mu,\ \bar{\mu} \notin B}{d \setminus B \xrightarrow{\langle \mu,t,r\rangle} d' \setminus B} \qquad Rel \quad \frac{d \xrightarrow{\langle \mu,t,r\rangle} d'}{d[\Phi] \xrightarrow{\langle \Phi(\mu),t,r\rangle} d'[\Phi]}$$

term $t \Rightarrow p[\text{rec } x.\ p/x]$ may also need applications of the clock distribution equations to become a state. Rule *Synch*, however, needs more explanation. It implements the so-called *busy-waiting* synchronisation mechanism according to which two parallel components can synchronise if they can perform communicating actions at the same time: if one of the two is able to execute such an action before the other, then a form of busy-waiting is allowed. However, when both partners are ready to synchronise, the handshaking immediately happens. Intuitively, this permits the modelling of the situation in which a faster process can wait for a slower partner. More formally, assume the left component d_1 completes the execution of action a at time instant t and with execution delay r_1 and the right component d_2 completes action $\bar{a}$ at time t and with execution delay r_2. Then, a synchronisation step is possible if and only if (at least) one of the two delays is 0, namely (at least) one of the two transitions is eager. The resulting transition is an invisible one, completing at time t and with execution delay 0[†].

[†] A different rule is used instead of this one in Aceto and Murphy (1996). Two processes can synchronise if they are ready to do so exactly at the same time instant: if a process P_1 can perform an action a at time n

https://doi.org/10.1017/S0960129503004018 Published online by Cambridge University Press

2.2. *Performance congruence*

Performance congruence (Corradini 1998) has been proposed in the field of semantics for process description languages as an equivalence notion that discriminates processes according to their functionality and performance. It refines *performance equivalence*, which had previously been proposed by Gorrieri and his co-authors (Gorrieri *et al.* 1995). Indeed, performance equivalence is not a congruence with respect to parallel composition with synchronisation. Performance congruence is preserved by every operator of the language and coincides with the largest congruence within performance equivalence.

Performance congruence is a bisimulation-based equivalence relation defined on top of the transition relation $d \xrightarrow{\langle \mu,t,r \rangle} d'$ given in Table 2. It relies on the same *busy-waiting* synchronisation mechanism of performance equivalence. In order to prove that two states d_1 and d_2 are *performance congruent*, it is required that if d_1 (and similarly for d_2) performs a visible action a at completion time t and with execution delay r, then d_2 can perform the same action at the same completion time and with arbitrary execution delay r'. This is called the lazy condition (Item (ii) in Definition 2.1). If $r = 0$, meaning that d_1 urgently performs action a, then d_2 is also forced to perform the same action in an eager way, that is, $r' = 0$. This is called the eager condition (Item (iii) in Definition 2.1). Invisible actions are always urgent as 'global actions'; namely, their execution delay is always 0 though (see Rule Synch in Table 2) a synchronisation can hide some delay.

Definition 2.1 (Performance congruence).

(1) A binary relation $\mathfrak{R}$ over $\mathscr{D}$ is a *PC-bisimulation* if and only if for each $(d_1, d_2) \in \mathfrak{R}$:

(i) (*Busy-Waiting*)
$d_1 \xrightarrow{\langle \tau,t,0 \rangle} d_1'$ implies $d_2 \xrightarrow{\langle \tau,t,0 \rangle} d_2'$ for some $d_2' \in \mathscr{D}$ such that $(d_1', d_2') \in \mathfrak{R}$.

(ii) (*Laziness*)
$d_1 \xrightarrow{\langle a,t,r \rangle} d_1'$ implies $d_2 \xrightarrow{\langle a,t,r' \rangle} d_2'$ for some $d_2' \in \mathscr{D}$, $r' \geqslant 0$ such that $(d_1', d_2') \in \mathfrak{R}$.

(iii) (*Eagerness*)
$d_1 \xrightarrow{\langle a,t,0 \rangle} d_1'$ implies $d_2 \xrightarrow{\langle a,t,0 \rangle} d_2'$ for some $d_2' \in \mathscr{D}$ such that $(d_1', d_2') \in \mathfrak{R}$.

(2) We say that two states d_1 and d_2 are *performance congruent*, $d_1 \sim_c d_2$, if and only if there exists a *PC-bisimulation* $\mathfrak{R}$ such that $(d_1, d_2) \in \mathfrak{R}$.

(3) We say that two processes p_1 and p_2 are *performance congruent*, $p_1 \sim_c p_2$, if and only if $0 \Rightarrow p_1 \sim_c 0 \Rightarrow p_2$.

The lazy and eager conditions in Definition 2.1 are needed to deal with invisible and visible actions in a consistent way. Invisible actions model synchronisations between two parallel components of the same global process ('internal' synchronisations). According to the busy-waiting synchronisation mechanism, a faster process can wait for a slower partner and no further delay is allowed when the two partners are ready to synchronise.

and a process P_2 can perform an action $\bar{a}$ at the same time, then the parallel composition of P_1 and P_2 can perform a synchronisation at time n. In this setting actions cannot be delayed – they must be performed as soon as they become available. Thus, this rule is suitable for modelling urgency in timed systems.

https://doi.org/10.1017/S0960129503004018 Published online by Cambridge University Press

Thus, as already remarked, a form of execution delay is possible to allow a synchronisation between two parallel components of the same global process. On the other hand, visible actions model synchronisations with the external environment ('external' synchronisations). They can be performed with an arbitrary delay before their execution (lazy transitions, Item (ii) in Definition 2.1) or with null execution delay (eager transitions, Item (iii) in Definition 2.1). The former executions model the situation in which the process responsible for their execution is faster with respect to a hypothetical external slower partner. The latter model the inverse situation, in which the process responsible for their execution is slower with respect to a faster external partner[†].

To study the discriminating power of eager, lazy and busy-waiting actions in our timed calculi, we consider two other equivalence relations:

— *Eager equivalence* (Gorrieri *et al.* 1995), denoted $\sim_e$, is obtained by removing item (ii) from Definition 2.1.
— *Lazy equivalence*, denoted $\sim_l$, is obtained by removing item (iii) from Definition 2.1.

3. Discriminating power of eager, lazy and busy-waiting actions

Consider the following restrictions over the base language $\mathscr{P}$ (similar restrictions and notation apply for $\mathscr{Q}$):

$\mathscr{P}_c$ – *The language that contains processes where choices are made at the same time,*
$\mathscr{P}_r$ – *The language that contains processes where relabelling functions are duration preserving,*
$\mathscr{P}_a$ – *The language that contains processes that can perform finitely many actions at a fixed time,*
$\mathscr{P}_v$ – *The language that contains processes that can perform visible actions.*

For any sequence $w = x_1x_2x_3\ldots \in \{c,r,a,v\}^+$, $\mathscr{P}_w$ denotes $\mathscr{P}_{x_1} \cap \mathscr{P}_{x_2} \cap \mathscr{P}_{x_3} \ldots$ and $\mathscr{D}_w$ denotes the set of timed states of processes in $\mathscr{P}_w$.

In the remainder of this section we will prove that the violation of one of these restrictions makes eager, lazy and busy-waiting actions more or less discriminating. This study is conducted by contrasting performance congruence, eager equivalence and lazy equivalence over the four different languages.

One result is simple. Performance congruence is always finer than both eager equivalence and lazy equivalence (see their definitions).

Proposition 3.1. Let $p_1, p_2 \in \mathscr{P}$. Then $p_1 \sim_c p_2$ implies $p_1 \sim_l p_2$ and $p_1 \sim_e p_2$.

In the following three subsections we show that lazy equivalence and eager equivalence are unrelated when the language takes into account choices at different time or non-duration preserving relabelling functions or processes that can perform infinitely many

[†] A similar distinction between 'lazy and eager observations' is also present in (the rather different) setting of the *TCSP*-based 'Time Stability Model' proposed in Reed and Roscoe (1988). In this timed model the basis actions are observed in two different ways: a denotes the communication of action a at any time while â denotes the communication of action a at the moment it becomes available.

https://doi.org/10.1017/S0960129503004018 Published online by Cambridge University Press

Table 3. *Delay Distribution Equations.*

$(n+m)p$	$=$	$(n)(m)p$
$(n)(p_1 \mid p_2)$	$=$	$(n)p_1 \mid (n)p_2$
$(n)(p_1 + p_2)$	$=$	$(n)p_1 + (n)p_2$
$(n)(p \setminus B)$	$=$	$(n)p \setminus B$
$(n)(p[\Phi])$	$=$	$(n)p[\Phi]$
rec $x.(n)s$	$=$	$(n)(s\{\text{rec } x.(n)s/x\})$

actions at the same time. In such cases, lazy equivalence and eager equivalence are strictly weaker than performance congruence by Proposition 3.1.

3.1. *Choosing at the same time*

We distinguish between languages that allow different alternatives to be chosen at different times or only at the same time. More precisely, if p_1 and p_2 are processes without delay operators at the top level, 'alternative timed compositions' are of the form

$$(t_1)p_1 \; + \; (t_2)p_2,$$

where t_1 and t_2 can be different[†]. 'Timed alternative compositions' are, by contrast, of the form

$$(t)(p_1 + p_2)$$

(possibly with applications, from right to left, of the rules in Table 3).

These two choice operators are conceptually different. They can be distinguished from a timing point of view. In $(t)(p_1 + p_2)$ the choice only involves the functionality of the system (the choice between p_1 and p_2), whereas in $(t_1)p_1 + (t_2)p_2$, the choice involves timed alternatives (timed functionalities) of the system.

Let $\cong$ be the least congruence that holds the laws in Table 3, and $\mathscr{S} \subseteq \mathscr{P}$ (ranged over by $s_1, s_2, \ldots$) be the set of closed terms generated by the following grammar (terms without delays operators at the top level):

$$s ::= nil \;\Big|\; a.q \;\Big|\; \sum_{i\in I} s_i \;\Big|\; \prod_{i\in I} s_i \;\Big|\; s \setminus B \;\Big|\; s[\Phi] \;\Big|\; x \;\Big|\; \text{rec } x.s$$

Then, we say that a choice $\sum_{i\in I} p_i$ is at the same time when either $\sum_{i\in I} p_i \in \mathscr{S}$ or $\sum_{i\in I} p_i \cong (n)\sum_{i\in I} s_i$ for some $n \in \mathbb{N}^+$; otherwise, $\sum_{i\in I} p_i$ is at different times.

The next propositions show that lazy equivalence and eager equivalence are unrelated when choices at different times are taken into account.

Proposition 3.2. If processes with choices at different times are allowed, there are p_1 and p_2 in $\mathscr{P}$ such that $p_1 \sim_l p_2$ does not imply $p_1 \sim_e p_2$.

[†] This non-deterministic choice operator behaves as the weak non-deterministic choice $\oplus$ in *TCCS* (Moller and Tofts 1990).

https://doi.org/10.1017/S0960129503004018 Published online by Cambridge University Press

Proof. Consider the following pair of processes:

$$p_1 = a + (k)a \quad \text{and} \quad p_2 = a$$

where $k \in \mathbb{N}^+$. They are lazy equivalent since each transition out of the right-hand side addend of p_1 can be matched by delayed transitions out of p_2. They are not eager equivalent, since p_1 can perform an eager transition at time $k + f(a)$ while p_2 cannot.

More generally, $p_1 = a + \sum_{i \in \mathbb{N}^+} (i)a$ is $\sim_l$-equivalent, but not $\sim_e$-equivalent, to p_2. □

Proposition 3.3. If processes with choices at different times are allowed, there are p_1 and p_2 in $\mathscr{P}$ such that $p_1 \sim_e p_2$ does not imply $p_1 \sim_l p_2$.

Proof. Consider the following pair of states:

$$p_1 = \ : (a \,|\, (k)b) + a.b \quad \text{and} \quad p_2 = a \,|\, (k)b$$

where $k \in \mathbb{N}^+$ is such that $k = f(a)$. They are eager equivalent since each transition out of the left-hand side addend of p_1 can be matched by a corresponding transition out of p_2. Moreover, they are not lazy equivalent. Choose $r > 0$. Then it is easy to convince oneself that lazy transition

$$(0 \Rightarrow p_1) \xrightarrow{\langle a, f(a)+r, r \rangle} (f(a) + r \Rightarrow b)$$

cannot be matched by p_2. □

3.2. *Relabelling by preserving action duration*

We distinguish between languages with relabelling functions that do not preserve the duration of the actions (for example, $\Phi(a) = b$ with $f(a) \neq f(b)$ is allowed), and languages with duration preserving relabelling functions (that is, $f(a) = f(\Phi(a))$ for every $a \in Act$).

If non-duration preserving relabelling functions are taken into account, lazy equivalence and eager equivalence are unrelated.

Proposition 3.4. If processes with non-duration preserving relabelling functions are allowed, there are p_1 and p_2 in $\mathscr{P}$ such that $p_1 \sim_l p_2$ does not imply $p_1 \sim_e p_2$.

Proof. Consider two actions $a, b \in Act$ such that $f(a) < f(b)$, and a relabelling function Φ (non-duration preserving such that $\Phi(a) = a$, $\Phi(b) = a$). Moreover, let

$$p_1 = (a + b)[\Phi] \quad \text{and} \quad p_2 = a.$$

Clearly, $p_1 \sim_l p_2$. However, $p_1 \not\sim_e p_2$ since p_1 can perform an eager a-action at time $f(b)$ whereas p_2 can only perform the same action at time $f(a) < f(b)$. □

Proposition 3.5. If processes with non-duration preserving relabelling functions are allowed, there are p_1 and p_2 in $\mathscr{P}$ such that $p_1 \sim_e p_2$ does not imply $p_1 \sim_l p_2$.

Proof. Consider actions $a, b, c \in Act$ such that $f(c) = f(a) + f(b)$, and a relabelling function Φ such that $\Phi(a) = a$, $\Phi(b) = b$, $\Phi(c) = b$. Let p_1 and p_2 be the pair of processes

https://doi.org/10.1017/S0960129503004018 Published online by Cambridge University Press

defined by

$$p_1 = (a.b + (a \mid c))[\Phi] \quad \text{and} \quad p_2 = (a \mid c)[\Phi].$$

In order to prove $p_1 \sim_e p_2$ the critical case is when the left-hand side addend of $(0 \Rightarrow p_1)$ performs an a-action. Transition $(0 \Rightarrow p_1) \xrightarrow{\langle a, f(a), 0\rangle} (f(a) \Rightarrow b)[\Phi]$ can only be matched by $(0 \Rightarrow p_2)$ performing $(0 \Rightarrow p_2) \xrightarrow{\langle a, f(a), 0\rangle} (f(a) \Rightarrow nil \mid 0 \Rightarrow c)[\Phi]$.

These two target states are clearly eager equivalent. If these two transitions are delayed $(r > 0)$,

$$(0 \Rightarrow p_1) \xrightarrow{\langle a, f(a)+r, r\rangle} (f(a) + r \Rightarrow b)[\Phi]$$

and

$$(0 \Rightarrow p_2) \xrightarrow{\langle a, f(a)+r, r\rangle} (f(a) + r \Rightarrow nil \mid 0 \Rightarrow c)[\Phi]$$

we obtain two target states that are not lazy congruent. Indeed, the former can perform a b-action at time $f(a) + r + f(b)$, whereas the latter can only perform the same action at time $f(c) = f(a) + f(b)$. Hence, $p_1 \not\sim_l p_2$. □

3.3. *Performing finitely many actions at the same time*

We distinguish between languages with processes that are able to perform infinitely many visible actions at a fixed time and languages with processes that are able to perform only finitely many visible actions at a fixed time (in the rest of the paper we will drop the word 'visible'). As an example, consider processes

$$p = \prod_{i \in \mathbb{N}} \{p_i = a\} \qquad \text{and} \qquad q = \sum_{i \in \mathbb{N}} \underbrace{a \mid \ldots \mid a}_{i \text{ times}},$$

and note that, when starting at time 0, process p can perform an infinite sequence of a-actions at time $f(a)$, whereas process q can only perform finite sequences of a-actions (although of unbounded length) at the same time.

Processes with infinitely many actions at a given time can be defined in two ways:

(a) *Unguarded Recursion.* That is, a variable x in a rec $x.p$ term can appear outside the scope of an $a.(\_)$ prefix operator. For instance, process rec $x.(x \mid a.nil)$ uses unguarded recursion to generate infinite concurrent a-actions at time $f(a)$ by assuming that the execution starts at time 0.

(b) *Infinite Parallel Composition.* That is, processes of the form $\prod_{i \in I} p_i$, where I can be infinite.

We now prove that lazy equivalence and eager equivalence are unrelated when unguarded recursion or infinite parallel composition are allowed.

Note that the proofs rely strongly on the fact that processes can perform infinitely many actions at a given time, independent of the fact that they are generated by unguarded recursion or infinite parallel composition. Thus, we will use p_∞ to denote a generic process that can generate infinitely many actions labelled with a at time $f(a)$, when starting at time 0. It can be either process $p_r = \text{rec } x.(x \mid a.nil)$ (in the case of unguarded recursion) or process $p_s = \prod_{i \in I} \{p_i = a\}$ with I infinite set (in the case of infinite parallel composition).

https://doi.org/10.1017/S0960129503004018 Published online by Cambridge University Press

Proposition 3.6. If processes with infinitely many actions at a given time are allowed, there are p_1 and p_2 in $\mathscr{P}$ such that $p_1 \sim_l p_2$ does not imply $p_1 \sim_e p_2$.

Proof. Processes p_1 and p_2 defined as

$$p_1 = b.a \,|\, p_\infty \quad \text{and} \quad p_2 = b \,|\, p_\infty$$

are lazy equivalent, but not eager equivalent. To prove $p_1 \sim_l p_2$, the only critical case is just when a b-action is performed. Suppose

$$(0 \Rightarrow p_1) \xrightarrow{\langle b, f(b)+r, r\rangle} d_1 = (f(b) + r \Rightarrow a) \,|\, (0 \Rightarrow p_\infty).$$

This transition can be matched by performing

$$(0 \Rightarrow p_2) \xrightarrow{\langle b, f(b)+r, r\rangle} d_2 = (f(b) + r \Rightarrow nil) \,|\, (0 \Rightarrow p_\infty).$$

States d_1 and d_2 are lazy equivalent. Again, the only critical case is when the left most component of d_1 performs the a-action. This move is easily matched by a delayed transition out of d_2. In particular, if the a-action performed by d_1 is eager, transition

$$d_1 \xrightarrow{\langle a, f(b)+r+f(a), 0\rangle} (f(b) + r + f(a) \Rightarrow nil) \,|\, (0 \Rightarrow p_\infty)$$

cannot be matched by a corresponding eager transition out of d_2. Hence, $p_1 \not\sim_e p_2$. □

Proposition 3.7. If processes with infinitely many actions at a given time are allowed, there are p_1 and p_2 in $\mathscr{P}$ such that $p_1 \sim_e p_2$ does not imply $p_1 \sim_l p_2$.

Proof. Processes p_1 and p_2 defined as

$$p_1 = (b + b.p_\infty) \,|\, b.p_\infty \quad \text{and} \quad p_2 = b \,|\, b.p_\infty$$

are eager equivalent, but not lazy equivalent.

In order to prove that $p_1 \sim_e p_2$, the only critical case is when the sub-component $b.p_\infty$ of $(b + b.p_\infty)$ in p_1 performs the b-action – namely,

$$(0 \Rightarrow p_1) \xrightarrow{\langle b, f(b), 0\rangle} d_1 = (f(b) \Rightarrow p_\infty) \,|\, (0 \Rightarrow b.p_\infty).$$

Then p_2 matches this transition with the following move:

$$(0 \Rightarrow p_2) \xrightarrow{\langle b, f(b), 0\rangle} d_2 = (0 \Rightarrow b) \,|\, (f(b) \Rightarrow p_\infty).$$

It is easy to prove that d_1 and d_2 are eager equivalent. However, if p_1 performs the same action in a lazy way,

$$(0 \Rightarrow p_1) \xrightarrow{\langle b, f(b)+r, r\rangle} d_1' = (f(b) + r \Rightarrow p_\infty) \,|\, (0 \Rightarrow b.p_\infty)$$

with $r > 0$, then the only reasonable transition p_2 can perform is

$$(0 \Rightarrow p_2) \xrightarrow{\langle b, f(b)+r, r\rangle} d_2' = (0 \Rightarrow b) \,|\, (f(b) + r \Rightarrow p_\infty).$$

States d_1' and d_2' cannot be lazy equivalent since after matching

$$d_1' \xrightarrow{\langle b, f(b), 0\rangle} d_1'' = (f(b) + r \Rightarrow p_\infty) \,|\, (f(b) \Rightarrow p_\infty)$$

https://doi.org/10.1017/S0960129503004018 Published online by Cambridge University Press

with

$$d_2' \xrightarrow{\langle b,f(b),0\rangle} d_2'' = (f(b) \Rightarrow nil) \,|\, (f(b) + r \Rightarrow p_\infty),$$

target state d_1'' can perform an eager a-action at time $f(b) + f(a)$, but d_2'' cannot. □

If unguarded recursion and infinite parallel composition are forbidden, our processes can only perform finitely many actions at a fixed time. The rest of this section is devoted to proving that if the language allows only

— finitely many actions to be performed at a given time,
— choices at the same time, and
— duration preserving relabelling functions,

then lazy equivalence coincides with performance congruence†, while eager equivalence still remains weaker than performance congruence. Thus, over the actual language, the lazy experiments have more discriminating power than the eager ones.

Theorem 3.1. Let $p_1, p_2 \in \mathscr{P}_{cra}$. Then $p_1 \sim_l p_2$ if and only if $p_1 \sim_c p_2$.

Proof. We will just give a sketch of the proof here – see Appendix A for a detailed proof. The *if* implication follows simply by Proposition 3.1. To prove the *only if* implication we show that every $\sim_l$-bisimulation is also a PC-bisimulation. The only critical case is eagerness of transitions, item (iii), since the laziness and the busy-waiting items, (i) and (ii), are common requirements of both $\sim_l$-bisimulation and PC-bisimulation. Hence, we have to prove that every transition with null execution delay out of a timed state can be matched by a transition with null execution delay out of the corresponding $\sim_l$-bisimilar timed state. The proof relies on the following two steps:

(a) Define a new bisimulation equivalence, called (n,t)-*performance congruence* and denoted $\sim_t^n$. This equivalence concentrates on visible actions with duration n and equates processes if and only if they can perform the same actions at time t. Its formal definition is similar to that of performance congruence when transitions are of the form $d \xrightarrow{\langle a,t,r\rangle} d'$, where a is such that $f(a) = n$. In the current language $\sim_t^n$ holds the following properties:

(1) (n,t)-*performance congruence* is a congruence;

(2) $\forall n \in \mathbb{N}^+$ and $\forall t \in \mathbb{N}$, $d_1 \sim_l d_2$ implies $d_1 \sim_t^n d_2$;

(3) $d_1 \xrightarrow{\langle a,t,r\rangle} d_2$ and $r > 0$ imply $d_1 \not\sim_{t-r}^n d_2$;

(4) $d_1 \xrightarrow{\langle a,t,0\rangle} d_2$ implies $d_1 \sim_{t^*}^n d_2$, for every $t^* < t$.

(b) Prove that if $d_1 \sim_l d_2$, then $d_1 \xrightarrow{\langle a,t,0\rangle} d_1'$ implies $d_2 \xrightarrow{\langle a,t,0\rangle} d_2'$ (and symmetrically for d_2) and $d_1' \sim_l d_2'$. Since $d_1 \sim_l d_2$, we certainly have $d_1 \xrightarrow{\langle a,t,0\rangle} d_1'$ implies $d_2 \xrightarrow{\langle a,t,r\rangle} d_2'$ and $d_1' \sim_l d_2'$. For a contradiction, assume $r > 0$. Then, item (2) implies $d_1 \sim_{t-r}^n d_2$ and $d_1' \sim_{t-r}^n d_2'$ implies $d_1 \sim_{t-r}^n d_1'$ by item (4). Item (1) (and, in particular, transitivity) implies $d_2 \sim_{t-r}^n d_2'$. This contradicts item (3), which, instead, states $d_2 \not\sim_{t-r}^n d_2'$. □

† This result was proved in Corradini and Di Cola (2001) over a language without delay operators.

https://doi.org/10.1017/S0960129503004018 Published online by Cambridge University Press

Proposition 3.8. Let $p_1, p_2 \in \mathscr{P}_{cra}$. Then $p_1 \sim_e p_2$ does not imply $p_1 \sim_c p_2$.

Proof. Consider the pair of processes p_1 and p_2 defined as

$$p_1 = (c.a \mid \bar{a} \mid b.\bar{a}) \setminus \{a\} \quad \text{and} \quad p_2 = (c.a \mid \bar{a} \mid b) \setminus \{a\},$$

where each action has the same duration, k say. They are eager equivalent but not performance congruent. The more involved case when proving $p_1 \sim_e p_2$ is when p_1 and p_2 perform the b-action followed by the c-action or vice versa. After these transitions the target states are $d_1 = (k \Rightarrow a \mid 0 \Rightarrow \bar{a} \mid k \Rightarrow \bar{a}) \setminus \{a\}$ and $d_2 = (k \Rightarrow a \mid 0 \Rightarrow \bar{a} \mid k \Rightarrow nil) \setminus \{a\}$, respectively. Both d_1 and d_2 can only perform a τ-move at time $2k$. Hence, p_1 and p_2 are eager equivalent. One can easily see that p_1 and p_2 cannot be performance congruent by performing the b-action with delay. □

From the previous result and the coincidence between performance congruence and lazy equivalence (Theorem 3.1), we have that eager equivalence does not imply lazy equivalence.

3.4. *Performing visible actions*

We distinguish between languages where process synchronisation is allowed and languages where process synchronisation is forbidden.

The language presented in Section 2 allows process synchronisation. To avoid process synchronisation, we can either remove the rule *Synch* or restrict the set of basic actions *Act* to A (in this way prefixes of the form $\bar{a}.(\_)$ are not allowed and the rule *Synch* in Table 2 never applies when one derives the transitional semantics of a process term). In both cases, the restrictions on the syntax of states and the transitional semantics are as expected.

Since the proofs given in the previous sections do not depend on invisible actions (apart from Proposition 3.8 for which we will get a surprising result when invisible actions are forbidden), they can be exploited when the actual language allows visible actions only. Thus, the main result of this section states that when the language allows only

— visible actions,
— finitely many actions to be performed at a given time,
— choices at the same time, and
— duration preserving relabelling functions,

performance congruence, eager equivalence and lazy equivalence coincide. Hence, in this case, testing for both eagerness and laziness (as performance congruence does) does not add any new insight compared with just testing for either eagerness or laziness separately. Similarly, since eager equivalence and lazy equivalence coincide, eager and lazy experiments have the same discriminating power. This latter result says, in other words, that when experimenting over processes we have two 'equivalent' ways to proceed: step-by-step (eager experiments) or jumping through time (lazy experiments).

We state the coincidence between performance congruence and eager equivalence (the coincidence between performance congruence and lazy equivalence was proved in Theorem 3.1).

https://doi.org/10.1017/S0960129503004018 Published online by Cambridge University Press

Theorem 3.2. Let $p_1, p_2 \in \mathscr{P}_{crav}$. Then $p_1 \sim_e p_2$ if and only if $p_1 \sim_c p_2$.

Proof. We will just give a sketch of the proof here – see Appendix B for a detailed proof. The *if* implication follows by Proposition 3.1. The *only if* implication follows from proving that a suitable $\sim_e$-bisimulation is also a PC-bisimulation. This means we have to prove that every transition with execution delay greater than 0 out of a timed state has to be matched by a transition with arbitrary execution delay out of the corresponding $\sim_e$-bisimilar timed state. This is the only critical case since eagerness of transitions, item (iii), is a common requirement of both $\sim_e$-bisimulation and PC-bisimulation. The proof relies on the following main, more involved, statements:

(a) If two $\mathscr{D}$ states, d' and d'', are eager equivalent, then there exists a pairwise eager equivalent decomposition of their parallel components. In other words, let $\equiv$ denote the congruence induced by the commutative and associative properties, the existence of unit object of parallel composition and the distribution of relabelling and restriction over parallel composition (remember that this language does not allow synchronisation). Then we have $d' \equiv (\Pi_{i\in I}\,(t_i \Rightarrow p_i))$, $d'' \equiv (\Pi_{i\in I}\,(t_i \Rightarrow q_i))$ and for each $i \in I$ it is $t_i \Rightarrow p_i \sim_e t_i \Rightarrow q_i$ (where the various p_i and q_i are in $\mathscr{S}$)[†].

(b) Let $t \Rightarrow p \sim_e t \Rightarrow q$ for some $t > 0$. Then, for every $t^* > t$, it is $t^* \Rightarrow p \sim_e t^* \Rightarrow q$.

(c) $\sim_e$ is preserved by every operator of the actual language.

The above statements are enough to prove that the relation

$$\mathfrak{R} = \{(d', d'') \mid d' \equiv (\Pi_{i\in I}\,(t_i \Rightarrow p_i)),\ d'' \equiv (\Pi_{i\in I}\,(t_i \Rightarrow q_i)),\ I = \{1, \ldots, n\} \text{ such that } n \in \mathbb{N}^+ \text{ and, for each } i \in I, (t_i \Rightarrow p_i) \sim_e (t_i \Rightarrow q_i),\ t_i \in \mathbb{N},\ p_i, q_i \in \mathscr{S}\}$$

is a PC-bisimulation. □

It is worth noting that eager equivalence is not preserved by parallel composition with synchronisation when invisible actions are taken into account.

Remark 3.1. Consider the pair of processes

$$p_1 \;=\; (c.\text{wait } 3.a.b \mid c.\text{wait } 3.a.b \mid \bar{a})\backslash\{a\}^{\ddagger}$$

and

$$p_2 \;=\; (c.\text{wait } 3.a.b \mid c.(\text{wait } 3.a.b + a.\text{wait } 3.b) \mid \bar{a})\backslash\{a\},$$

and assume that the duration of a, b and c is 3 (as well as the duration of co-actions $\bar{a}$ and $\bar{c}$). In Corradini (1998) it was proved that $p_1 \sim_e p_2$ and that $p_1 \mid \bar{c}.\bar{c} \not\sim_e p_2 \mid \bar{c}.\bar{c}$.

We conclude this section by showing that lazy equivalence is not preserved by parallel composition with synchronisation when the language allows, respectively, only choices at

[†] A similar decomposition lemma has been fruitful in Bérard *et al.* (2000) for deciding eager equivalence in polynomial time.

[‡] wait $t.\,p$ is an abbreviation for a process that evolves into p after performing an internal action taking t time units. It is possible to think of wait $t.\,p$ as an abbreviation for $(a|\bar{a}.p)\backslash\{a\}$ (where a is not free in p and $f(a) = t$).

https://doi.org/10.1017/S0960129503004018 Published online by Cambridge University Press

different times (Remark 3.2), relabelling functions non-duration preserving (Remark 3.3) and infinitely many actions at a fixed time (Remark 3.4).

Remark 3.2. Consider the pair of processes $p_1 = a + (k)a$ and $p_2 = a$ given in Proposition 3.2 ($k \in \mathbb{N}^+$). It has been shown that $p_1 \sim_l p_2$. Now consider a third process $p_3 = \bar{a}$. The parallel compositions $p_1 \mid p_3$ and $p_2 \mid p_3$ are not lazy equivalent since the former can perform a τ-action at time $k + f(a)$ while the latter can only do the same at time $f(a)$.

Remark 3.3. Consider the pair of processes $p_1 = (a + b)[\Phi]$ and $p_2 = a$ where actions $a, b \in Act$ are such that $f(a) < f(b)$. Moreover, assume that $\Phi(a) = a$, $\Phi(b) = a$. In Proposition 3.4 we have shown that they are such that $p_1 \sim_l p_2$. However, $p_1 \mid p_3$ and $p_2 \mid p_3$, where $p_3 = \bar{a}$ are such that $p_1 \mid p_3 \not\sim_l p_2 \mid p_3$, since the former can perform a τ-action at time $f(b)$ or at time $f(a)$, while the latter can only perform τ at time $f(a) < f(b)$.

Remark 3.4. Consider the pair of processes $p_1 = b.a \mid p_\infty$ and $p_2 = b \mid p_\infty$ given in Proposition 3.6. They are such that $p_1 \sim_l p_2$. Now, consider process $p_3 = \bar{a}$ and the parallel compositions $p_1 \mid p_3$ and $p_2 \mid p_3$. We have $p_1 \mid p_3 \not\sim_l p_2 \mid p_3$. Indeed, consider the b-transitions

$$0 \Rightarrow (p_1 \mid p_3) \xrightarrow{\langle b, f(b), 0 \rangle} d_1 = (f(b) \Rightarrow a) \mid (0 \Rightarrow p_\infty) \mid (0 \Rightarrow \bar{a})$$
$$0 \Rightarrow (p_2 \mid p_3) \xrightarrow{\langle b, f(b), 0 \rangle} d_2 = (f(b) \Rightarrow nil) \mid (0 \Rightarrow p_\infty) \mid (0 \Rightarrow \bar{a}).$$

The target states d_1 and d_2 are such that $d_1 \not\sim_l d_2$ since the former can perform a τ-action at time $f(a) + f(b)$, while the latter cannot.

The intuition behind these negative congruence results is that when busy-waiting process synchronisation is allowed but either item (ii) or item (iii) is removed from the definition of performance congruence, the resulting equivalences deal with 'internal' synchronisations and 'external' synchronisations in a non-consistent way (recall the discussion after Definition 2.1). Because of this inconsistent treatment of internal and external synchronisations, eager equivalence (performance equivalence) and lazy equivalence have the unfortunate effect of losing compositionality. They are not preserved by parallel composition with synchronisation, as the above remarks have shown.

4. Concluding remarks and related work

We have investigated the discriminating power of timed actions in timed computation. Such power depends on the language for the description of time-critical systems. We have detected four significant language features that give rise to different languages and make three performance-sensitive equivalences (performance congruence, lazy equivalence and eager equivalence) behave differently over these languages. Table 4 summarises our results.

Note that if process synchronisation is allowed and eager equivalence and/or lazy equivalence do not coincide with performance congruence, they are not even compositional. In particular, besides being unrelated equivalences, they are not even congruences for parallel composition with synchronisation. Of course, congruence properties of process equivalences are of great benefit during the verification phase of concurrent and distributed

https://doi.org/10.1017/S0960129503004018 Published online by Cambridge University Press

Table 4. *Relationships among the three equivalences.*

Choosing at the same time	Relabelling by preserving action duration	Performing finitely many actions at the same time	Performing only visible actions	Results
No	Yes/No	Yes/No	Yes/No	$\sim_c \subsetneq \sim_l$ $\sim_c \subsetneq \sim_e$ $\sim_e \neq \sim_l$
Yes/No	No	Yes/No	Yes/No	$\sim_c \subsetneq \sim_l$ $\sim_c \subsetneq \sim_e$ $\sim_e \neq \sim_l$
Yes/No	Yes/No	No	Yes/No	$\sim_c \subsetneq \sim_l$ $\sim_c \subsetneq \sim_e$ $\sim_e \neq \sim_l$
Yes	Yes	Yes	No	$\sim_c = \sim_l$ $\sim_c \subsetneq \sim_e$ $\sim_l \subsetneq \sim_e$
Yes	Yes	Yes	Yes	$\sim_l = \sim_c = \sim_e$

systems. Thus, our work also shows how congruence properties of performance-sensitive equivalences may depend on the expressive power of the language. As a general result, we have that the three items in the definition of performance congruence are the required ingredients for capturing the coarsest congruence within the performance equivalence in Gorrieri *et al.* (1995) (Corradini 1998).

The present work is related to Corradini *et al.* (to appear) and Corradini and Di Cola (2001). The former develops a mathematical framework for describing and reasoning about semantic theories for processes with durational actions. The comparison, however, only involves the semantic theories in the sense that a common language is considered for all of them. The latter mainly proves Theorem 3.1, which, in turn, solves a conjecture in Corradini (1998). Here, we have extended that work by adding eager equivalence to the comparison between lazy equivalence and performance congruence, facilities for process delay and some more significant language features. In general, the addition of delayed processes complicates the proofs of our statements considerably.

Appendix A. A proof of Theorem 3.1

The proof of Theorem 3.1 requires some new notions and technical results. For the sake of comprehensiveness, this section is divided into three subsections.

Recall that in this section the current language is $\mathscr{P}_{cra}$; the set of $\mathscr{P}$ processes that allow finitely many actions to be performed at a given time, choices at the same time and duration preserving relabelling functions. Invisible actions are taken into account. Since

https://doi.org/10.1017/S0960129503004018 Published online by Cambridge University Press

the language in Appendix B is a sublanguage of $\mathscr{P}_{cra}$, some results stated here will be exploited to prove Theorem 3.2 (in Appendix B).

Remark A.1. We say that a choice $\sum_{i\in I} p_i$ is at the same time when either $\sum_{i\in I} p_i \in \mathscr{S}$ or $\sum_{i\in I} p_i \cong (n)\sum_{i\in I} s_i$ for some $n \in \mathbb{N}^+$; otherwise, $\sum_{i\in I} p_i$ is at different times (see Section 3.1). Anyway, in this section (and Appendix B) we will only deal with summations of the form $\sum_{i\in I} p_i \in \mathscr{S}$. Summations of the form $\sum_{i\in I} p_i \cong (n)\sum_{i\in I} s_i$ can be dealt with in the usual way. Note that the rules in Table 3 are sound with respect to all the equivalences we consider, and these equivalences are congruences over the actual languages. Thus we can assume that every occurrence of a summation $\sum_{i\in I} p_i$ is replaced in a process term with the $\cong$-equivalent one $(n)\sum_{i\in I} s_i$. Then every $(n)\sum_{i\in I} s_i$ term can be thought of as the composition of a delay operator $(n)\_$ and a summation $\sum_{i\in I} s_i$ (where every s_i does not have delay operators at the top level). Both of them are primitive operators in the language and will be dealt with separately in inductive proofs.

A.1. *Infinite computations*

First we prove that when unguarded recursion and infinite parallel composition are removed from the calculus, no state can perform infinite computations at a given time. Actually, we can prove more: each state cannot start the execution of infinitely many concurrent actions at the same time. The following definition makes this latter claim more precise.

We start by defining the derivatives of a state.

Definition A.1. We say that $d' \in \mathscr{D}_{cra}$ is a *derivative* of $d \in \mathscr{D}_{cra}$ if and only if either $d' = d$ or there exists a set of states $\{d_0, \ldots, d_k\}$ $(d_i \in \mathscr{D}_{cra})$ such that $d_0 = d$, $d_k = d'$ and $d_{i-1} \xrightarrow{\langle \mu_i, t_i, r_i\rangle} d_i$ for some $t_i, r_i \in \mathbb{N}$ and $\mu_i \in Act_\tau$ $(0 < i \leqslant k)$.

Then, we formalise when a $\mathscr{D}_{cra}$ state, or, equivalently, a $\mathscr{P}_{cra}$ process, has an infinite computation at (starting) time t.

Definition A.2. Let $t \in \mathbb{N}$.

— A state $d \in \mathscr{D}_{cra}$ has an *infinite computation at t time* if and only if there exist a derivative d' of d and an infinite set of states $\{d_0, \ldots, d_i, \ldots\}$ such that $d_0 = d'$, and for every $i \geqslant 0$ we have $d_i \xrightarrow{\langle a_i, t, r_i\rangle} d_{i+1}$, for some $a_i \in Act$ and $r_i \geqslant 0$.

— A state $d \in \mathscr{D}_{cra}$ has an *infinite computation at t starting time* if and only if there exist a derivative d' of d and an infinite set of states $\{d_0, \ldots, d_i, \ldots\}$ such that $d_0 = d'$, and for every $i \geqslant 0$ we have $d_i \xrightarrow{\langle a_i, t_i, r_i\rangle} d_{i+1}$ and $t_i - f(a_i) - r_i = t$ for some $a_i \in Act$ and $t_i, r_i \geqslant 0$.

— A process $p \in \mathscr{P}_{cra}$ has an *infinite computation at t (starting) time* if and only if $0 \Rightarrow p$ has an infinite computation at t (starting) time.

Terms in $\mathscr{D}_{cra}$ do not have infinite computations at t (starting) time. For standard reasons, we prove it for open terms. In the following lemma, given $q, q_1, \ldots, q_k \in \mathscr{Q}_{cra}$ and

https://doi.org/10.1017/S0960129503004018 Published online by Cambridge University Press

$x_1, \ldots, x_k \in Var$, we use $q[q_1/x_1, \ldots, q_k/x_k]$ to denote term q where every occurrence of x_i is replaced by q_i.

Lemma A.1. Let $q \in \mathscr{Q}_{cra}$ such that every free variable is guarded. Then, for every set $\{x_1, \ldots, x_k\} \supseteq \mathscr{F}(q)$, $\forall p_{x_1}, \ldots, p_{x_k} \in \mathscr{P}_{cra}$ and $\forall t, t' \in \mathbb{N}$, we have that the state $t' \Rightarrow q[p_{x_1}/x_1, \ldots, p_{x_k}/x_k]$ does not have infinite computations at t (starting) time.

Proof. Assume $q \in \mathscr{Q}_{cra}$ such that each free variable is guarded, $\{x_1, \ldots, x_k\} \supseteq \mathscr{F}(q)$, $p_{x_1}, \ldots, p_{x_k} \in \mathscr{P}_{cra}$, $t, t' \in \mathbb{N}$, and prove that state $t' \Rightarrow q[p_{x_1}/x_1, \ldots, p_{x_k}/x_k]$ does not have infinite computations at t time by induction on the syntactic structure of q (the case of infinite computations at t starting time follows by similar reasoning). Moreover, assume different names from $\{x_1, \ldots, x_k\}$ for variables appearing within q. Otherwise, apply alpha-conversion to avoid collision of names. This does not change the possibility of having infinite computations at t time. Let $t' < t$, since otherwise the proof is simple, indeed, $t' \Rightarrow q[p_{x_1}/x_1, \ldots, p_{x_k}/x_k]$ cannot perform any action in Act at time t.

The more involved case is when q is a recursive process. We only prove this case in detail because all the others are simpler and follow by simple inductive reasoning.

Assume $q = \text{rec } x.q_1$. Because we have assumed that q has different variable names from $\{x_1, \ldots, x_k\}$, we can suppose $x \notin \{x_1, \ldots, x_k\}$. Moreover, in spite of the fact that recursion is guarded, x is a free guarded variable for q_1. Hence, $\{x_1, \ldots, x_k, x\} \supseteq \mathscr{F}(q_1)$. For each $p \in \mathscr{P}_{cra}$, by the induction hypothesis we have that $t' \Rightarrow q_1[p_{x_1}/x_1, \ldots, p_{x_k}/x_k, p/x]$ does not have infinite computations at t time. Note that $t' \Rightarrow q[p_{x_1}/x_1, \ldots, p_{x_k}/x_k]$ coincides with

$$t' \Rightarrow \text{rec } x.(q_1[p_{x_1}/x_1, \ldots, p_{x_k}/x_k]).$$

Thus, if by contradiction this state had infinite computations at t time, state

$$t' \Rightarrow (q_1[p_{x_1}/x_1, \ldots, p_{x_k}/x_k])[\text{rec } x.\,(q_1[p_{x_1}/x_1, \ldots, p_{x_k}/x_k])/x],$$

which coincides with

$$t' \Rightarrow q_1[p_{x_1}/x_1, \ldots, p_{x_k}/x_k, \text{rec } x.(q_1[p_{x_1}/x_1, \ldots, p_{x_k}/x_k])/x],$$

would also have infinite computations at t time.

By taking $p = \text{rec } x.q_1[p_{x_1}/x_1, \ldots, p_{x_k}/x_k]$, we contradict the induction hypothesis. □

As a corollary of the previous lemma, we have a similar result in the class of closed $\mathscr{D}_{cra}$ states (and, hence, $\mathscr{P}_{cra}$ processes also).

Proposition A.1. Let $d \in \mathscr{D}_{cra}$. Then, for every $t \in \mathbb{N}$, state d does not have infinite computations at t (starting) time.

A.2. *Parallel contexts*

A *parallel context*, denoted by $C_{[]}$, is a term with a single 'hole' (single occurrence of []) of the following grammar:

$$C_{[]} ::= [] \;\Big|\; C_{[]} \,|\, d \;\Big|\; d \,|\, C_{[]} \;\Big|\; C_{[]} \setminus B \;\Big|\; C_{[]}[\Phi],$$

where d is a $\mathscr{D}_{cra}$ term.

https://doi.org/10.1017/S0960129503004018 Published online by Cambridge University Press

Given a state d and a parallel context $C_{[]}$, we use $C[d]$ to denote the state obtained by replacing [] with d in $C_{[]}$. We use C to denote the state $C[0 \Rightarrow nil]$.

The following lemma proves that whenever a state can perform a lazy (eager) transition at time t and delay $r > 0$ ($r = 0$), it can also perform a corresponding eager (lazy with arbitrary delay k) transition at time $t - r$ ($t + k$). Its proof follows by simple inductive reasoning.

Lemma A.2. Let $d_1, d_2 \in \mathscr{D}_{cra}$, $a \in Act$, $t, r \in \mathbb{N}$, $k \in \mathbb{N}^+$ and $d_1 \xrightarrow{\langle a,t,r\rangle} d_2$. Then, there are a parallel context $C_{[]}$ and a process $p \in \mathscr{P}_{cra}$ such that $d_2 = C[t \Rightarrow p]$ and:

(1) $d_1 \xrightarrow{\langle a,t-r,0\rangle} C[t - r \Rightarrow p]$, if $r > 0$ (*laziness vs. eagerness*)

(2) $d_1 \xrightarrow{\langle a,t+k,k\rangle} C[t + k \Rightarrow p]$, if $r = 0$ (*eagerness vs. laziness*).

A.3. *The main results*

We now define a new bisimulation equivalence, called (n,t)-*performance congruence* and denoted $\sim_t^n$. This equivalence concentrates on visible actions with duration n, and equates processes whenever they can perform the same actions at time t.

Definition A.3 ((n,t)-performance congruence).

(1) Let $n \in \mathbb{N}^+$ and $t \in \mathbb{N}$. The functional $PC_t^n : Rel \to Rel$ is defined, for each $\mathfrak{R} \in Rel$, as follows: $(d_1, d_2) \in PC_t^n(\mathfrak{R})$ if, for each $a \in Act(n)$,

— $d_1 \xrightarrow{\langle a,t,r\rangle} d_1'$ implies $d_2 \xrightarrow{\langle a,t,r'\rangle} d_2'$ for some $d_2' \in \mathscr{D}_{cra}$ and $r' \geqslant 0$ such that $(d_1', d_2') \in \mathfrak{R}$.

(2) A relation $\mathfrak{R} \in Rel$ will be called PC_t^n-*bisimulation* if $\mathfrak{R} \subseteq PC_t^n(\mathfrak{R})$.

(3) We say that two states d_1 and d_2 are (n,t)-*performance congruent*, $d_1 \sim_t^n d_2$, if and only if there exists a PC_t^n-bisimulation $\mathfrak{R}$ such that $(d_1, d_2) \in \mathfrak{R}$.

(4) We say that two processes p_1 and p_2 are (n,t)-*performance congruent*, $p_1 \sim_t^n p_2$, if and only if $(0 \Rightarrow p_1) \sim_t^n (0 \Rightarrow p_2)$.

Clearly, lazy equivalence is strictly finer than (n,t)-performance congruence, and $\sim_t^n$ is preserved by all operators of the language. The former statement follows by their definitions, while the latter follows on standard lines.

Proposition A.2. Let $p_1, p_2 \in \mathscr{P}_{cra}$ such that $p_1 \sim_l p_2$. Then, $p_1 \sim_t^n p_2$ for each $n \in \mathbb{N}^+$ and $t \in \mathbb{N}$.

A key lemma of this section states that the target state of a transition at time t over an action a out of a state d cannot be $(f(a),t)$-performance congruent to d. It will exploit the fact that states, in the current language, cannot perform infinite computations at time t (Proposition A.1).

Lemma A.3. Let $d_1, d_2 \in \mathscr{D}_{cra}$, $a \in Act$, $t \in \mathbb{N}$ and $d_1 \xrightarrow{\langle a,t,r_1\rangle} d_2$ with $r_1 \geqslant 0$. Then, $d_1 \not\sim_t^{f(a)} d_2$.

Proof. Assume $d_1, d_2 \in \mathscr{D}_{cra}$, $a \in Act$, $t, r_1 \in \mathbb{N}$ such that $d_1 \xrightarrow{\langle a,t,r_1\rangle} d_2$. Moreover, to facilitate the notation, consider $f(a) = n$. For a contradiction, suppose $d_1 \sim_t^n d_2$. Then,

https://doi.org/10.1017/S0960129503004018 Published online by Cambridge University Press

there exists a set of states $\{d_1, d_2, \ldots, d_i, \ldots\}$ such that

$$\begin{array}{lllll}
d_1 \xrightarrow{\langle a,t,r_1\rangle} d_2 & \text{and} & d_1 \sim_t^n d_2 & \Rightarrow & d_2 \xrightarrow{\langle a,t,r_2\rangle} d_3 \quad \text{and} \quad d_2 \sim_t^n d_3 \\
d_2 \xrightarrow{\langle a,t,r_2\rangle} d_3 & \text{and} & d_2 \sim_t^n d_3 & \Rightarrow & d_3 \xrightarrow{\langle a,t,r_3\rangle} d_4 \quad \text{and} \quad d_3 \sim_t^n d_4 \\
\vdots & & \vdots & & \vdots \\
d_i \xrightarrow{\langle a,t,r_i\rangle} d_{i+1} & \text{and} & d_i \sim_t^n d_{i+1} & \Rightarrow & d_{i+1} \xrightarrow{\langle a,t,r_{i+1}\rangle} d_{i+2} \quad \text{and} \quad d_{i+1} \sim_t^n d_{i+2} \\
\vdots & & \vdots & & \vdots
\end{array}$$

where r_i are proper execution delays. By Proposition A.1, there must exist $k \in \mathbb{N}$ such that state d_k cannot perform any action a at time t. Otherwise, d_1 would have an infinite computation at time t. However, k leads to a contradiction with the hypothesis because we would have $d_{k-1} \xrightarrow{\langle a,t,r_{k-1}\rangle} d_k$ and $d_{k-1} \sim_t^n d_k$, while, from the hypothesis, d_k cannot perform any action a at time t. Thus, states d_{k-1} and d_k cannot be (n,t)-performance congruent. □

On the other hand, if the transition out of d at time t over the action a is eager, the corresponding target state is $(f(a), t^*)$-performance congruent to d, provided that $t^* < t$.

Lemma A.4. Let $d_1, d_2 \in \mathscr{D}_{cra}$, $n \in \mathbb{N}^+$, $a \in Act(n)$ and $t \in \mathbb{N}$ such that $d_1 \xrightarrow{\langle a,t,0\rangle} d_2$. Then, $d_1 \sim_{t^*}^n d_2$ for each $t^* < t$.

Proof. Assume $d_1, d_2 \in \mathscr{D}_{cra}$, $n \in \mathbb{N}^+$, $a \in Act(n)$, $t \in \mathbb{N}$ and $t^* < t$ such that $d_1 \xrightarrow{\langle a,t,0\rangle} d_2$. We proceed by induction on the depth of transition $d_1 \xrightarrow{\langle a,t,0\rangle} d_2$ by case analysis on the structure of state d_1.

(a) Case $d_1 = t' \Rightarrow nil$ is impossible, for each $t' \in \mathbb{N}$.
(b) Case $d_1 = t' \Rightarrow b.p$ with $b \neq a$ and $t' \neq t - n$ is impossible. Assume $b = a$ with states $d_1 = t - n \Rightarrow a.p$ and $d_2 = t \Rightarrow p$. Neither state can perform any action in $Act(n)$ at time $t^* < t$. Hence, $d_1 \sim_{t^*}^n d_2$.
(c) Case $d_1 = t' \Rightarrow (m)p$ with $t' \neq t - m - n$ is impossible. Assume $d_1 = t - m - n \Rightarrow (m)p$. By the operational rules, $d_1 \xrightarrow{\langle a,t,0\rangle} d_2$ is derivable if and only if $t' + m \Rightarrow p \xrightarrow{\langle a,t,0\rangle} d_2$. By the induction hypothesis, $t' + m \Rightarrow p \sim_{t^*}^n d_2$. By considering the Clock Distribution Equations in Table 1, this also means $d_1 \sim_{t^*}^n d_2$.
(d) Let $d_1 = d_3 \mid d_4$ and assume $d_3 \mid d_4 \xrightarrow{\langle a,t,0\rangle} d_3' \mid d_4$ if $d_3 \xrightarrow{\langle a,t,0\rangle} d_3'$ (the other case is similar). By the induction hypothesis, $d_3 \sim_{t^*}^n d_3'$. Because (n,t)-performance congruence is preserved by parallel composition, we also have $d_3 \mid d_4 \sim_{t^*}^n d_3' \mid d_4$.
(e) Cases $d_1 = d_1' \setminus B$ and $d_1 = d_1'[\Phi]$ follow by similar reasoning to the previous one.
(f) Let $d_1 = t' \Rightarrow \sum_{i \in I} p_i$. Assume $\sum_{i \in I} p_i \in \mathscr{S}$ and $t' = t - f(a)$ (otherwise the proof would be trivial). By the operational rules, $d_1 \xrightarrow{\langle a,t,0\rangle} d_2$ is derivable if and only if $\exists j \in I$ such that $t' \Rightarrow p_j \xrightarrow{\langle a,t,0\rangle} d_2$. By the induction hypothesis, we must have that $t' \Rightarrow p_j \sim_{t^*}^n d_2$. By the transitivity of $\sim_{t^*}^n$, it is then sufficient to prove that $(t' \Rightarrow p_j) \sim_{t^*}^n (t' \Rightarrow \sum_{i \in I} p_i)$ to conclude $(t' \Rightarrow \sum_{i \in I} p_i) \sim_{t^*}^n d_2$. Indeed, since t' is the local clock of both $(t' \Rightarrow p_j)$ and $(t' \Rightarrow \sum_{i \in I} p_i)$, each transition performing actions in $Act(n)$ out of these two states is visible at completion time $t' + n = t$. By hypothesis, $t^* < t$, and they are

https://doi.org/10.1017/S0960129503004018 Published online by Cambridge University Press

(n,t^*)-performance congruent just because they cannot perform any action in $Act(n)$ at time t^*.

(g) Case $d_1 = (t' \Rightarrow \mathrm{rec}\ x.p)$ is similar to the previous one. □

The next lemma is related to lazy transitions. If the transition out of d at time t over the action a is lazy with delay $r > 0$, the corresponding target state cannot be $(f(a), t-r)$-performance congruent to d.

Lemma A.5. Let $d_1, d_2 \in \mathscr{D}_{cra}$, $a \in Act$, and $t \in \mathbb{N}$ be such that $d_1 \xrightarrow{\langle a,t,r\rangle} d_2$ with $r > 0$. Then, $d_1 \not\sim^{f(a)}_{t-r} d_2$.

Proof. Assume $d_1, d_2 \in \mathscr{D}_{cra}$, $a \in Act$, and $t \in \mathbb{N}$ such that $d_1 \xrightarrow{\langle a,t,r\rangle} d_2$ with $r > 0$. By Lemma A.2 applied to transition $d_1 \xrightarrow{\langle a,t,r\rangle} d_2$, there exist a parallel context $C_{[]}$ and a process p' such that $d_2 = C[t \Rightarrow p']$ and $d_1 \xrightarrow{\langle a,t-r,0\rangle} d_3 = C[t-r \Rightarrow p']$. By Lemma A.3 applied to this latter transition, $d_1 \not\sim^{f(a)}_{t-r} d_3$. To conclude the proof, we just have to show that $d_2 \sim^{f(a)}_{t-r} d_3$ (if for a contradiction, $d_1 \sim^{f(a)}_{t-r} d_2$, then by the transitivity of (n,t)-performance congruence, we also would have $d_1 \sim^{f(a)}_{t-r} d_3$). Hence, to show $d_2 \sim^{f(a)}_{t-r} d_3$, consider

$$\mathfrak{R} = \{(C'[t \Rightarrow p], C'[t-r \Rightarrow p]) \mid C'_{[]} \text{ is a parallel context and } p \in \mathscr{P}\}.$$

Clearly, $(d_2, d_3) \in \mathfrak{R}$. It remains to prove that $\mathfrak{R}$ is a $\mathrm{PC}^{(t-r)}_{f(a)}$-bisimulation. Consider pair $(d'_1, d'_2) \in \mathfrak{R}$ such that $(d'_1, d'_2) = (C'[t \Rightarrow p], C'[t-r \Rightarrow p])$ for some parallel context $C'_{[]}$ and process p. Assume $d'_1 \xrightarrow{\langle b,t-r,r^*\rangle} d''_1$ for some $b \in Act(f(a))$ and $r^* \geqslant 0$ (the other case is similar). Because $d'_1 = C'[t \Rightarrow p]$ and $t > t-r$, we must have $d''_1 = C''[t \Rightarrow p]$ with $C' \xrightarrow{\langle b,t-r,r^*\rangle} C''$, that is, component $t \Rightarrow p$ cannot perform any action in $Act(f(a))$ at time $t-r$. By similar reasoning, $d'_2 \xrightarrow{\langle b,t-r,r^*\rangle} d''_2 = C''[t-r \Rightarrow p]$ also. Hence, $(d''_1, d''_2) \in \mathfrak{R}$. □

Theorem 3.1. Let $p_1, p_2 \in \mathscr{P}_{cra}$. Then $p_1 \sim_l p_2$ if and only if $p_1 \sim_c p_2$.

Proof. Assume $p_1, p_2 \in \mathscr{P}_{cra}$. The fact that $p_1 \sim_c p_2$ implies $p_1 \sim_l p_2$ immediately follows by Proposition 3.1. Now, suppose $p_1 \sim_l p_2$ and prove $p_1 \sim_c p_2$. By hypothesis, there exists a $\sim_l$-bisimulation $\mathfrak{R}$ such that $(0 \Rightarrow p_1, 0 \Rightarrow p_2) \in \mathfrak{R}$. To conclude the proof it is sufficient to show that $\mathfrak{R}$ is also a PC-bisimulation. Consider pair $(d_1, d_2) \in \mathfrak{R}$. The laziness and busy-waiting items immediately follow by the fact that $\mathfrak{R}$ is a $\sim_l$-bisimulation. Let us prove the eagerness item. By hypothesis, $d_1 \sim_l d_2$, and if $d_1 \xrightarrow{\langle a,t,0\rangle} d'_1$ (the symmetric case is similar), we can derive $d_2 \xrightarrow{\langle a,t,r\rangle} d'_2$ with $r \geqslant 0$ and $(d'_1, d'_2) \in \mathfrak{R}$. We show that it must necessarily be $r = 0$. For a contradiction, suppose $r > 0$. By Lemma A.4 applied to transition $d_1 \xrightarrow{\langle a,t,0\rangle} d'_1$ with $t^* = t-r < t$, we have $d_1 \sim^{f(a)}_{t^*} d'_1$. Moreover, by $d_1 \sim_l d_2$, $d'_1 \sim_l d'_2$ and the fact that $\sim_l$ implies $\sim^n_t$ (Proposition A.2), we have $d_1 \sim^{f(a)}_{t^*} d_2$ and $d'_1 \sim^{f(a)}_{t^*} d'_2$. Hence, we obtain

$$d_2 \sim^{f(a)}_{t^*} d_1 \sim^{f(a)}_{t^*} d'_1 \sim^{f(a)}_{t^*} d'_2,$$

and, by the transitivity of $\sim^{f(a)}_{t^*}$, we have $d_2 \sim^{f(a)}_{t^*} d'_2$. This contradicts Lemma A.5 applied to transition $d_2 \xrightarrow{\langle a,t,r\rangle} d'_2$ with $r > 0$. □

https://doi.org/10.1017/S0960129503004018 Published online by Cambridge University Press

Appendix B. A proof of Theorem 3.2

The proof of Theorem 3.2 also requires some new notions and technical results. We follow a similar scheme to that of Appendix A by grouping related results/definitions into four subsections.

The language we consider in this section is $\mathscr{P}_{crav}$; the set of $\mathscr{P}$ processes that allow only visible actions, finitely many actions to be performed at a given time, choices at the same time, and duration preserving relabelling functions. Since the current language does not have invisible actions, we can exploit the results stated in Appendix A without problem. In particular, Proposition A.1 will be useful.

B.1. *Over-timing relation*

Consider $\mathbb{N}^\star = \mathbb{N} \cup \{+\infty, -\infty\}$ and the ordering on natural numbers extended by $-\infty < n < +\infty$ $(\forall n \in \mathbb{N})$.

Let $d \in \mathscr{D}_{crav}$ and $Y \in Act$ be a set of actions. Define function *minclk* that, given a state $d \in \mathscr{D}_{crav}$ and a set of actions $Y \subseteq Act$, returns the *minimum* local clock associated with the components of d that are able to perform at least an action in Y. A local clock with such a property will be called *active*.

Definition B.1. Let $minclk : \mathscr{D}_{crav} \times 2^{Act} \to \mathbb{N}^\star$ be the least relation that satisfies the following inference rules (where $t \in \mathbb{N}$, $a \in Act$, $Y \subseteq Act$, $d, d_1, d_2 \in \mathscr{D}_{crav}$ and $p_j, \text{rec } x.p \in \mathscr{P}_{crav}$):

$$
\begin{aligned}
minclk(t \Rightarrow nil,\ Y) &= +\infty \\
minclk(t \Rightarrow a.p,\ Y) &= \begin{cases} t & \text{if } a \in Y \\ +\infty & \text{otherwise} \end{cases} \\
minclk(t \Rightarrow (n)p,\ Y) &= minclk(t + n \Rightarrow p,\ Y) \\
minclk(t \Rightarrow \textstyle\sum_{i \in I} p_i,\ Y) &= \begin{cases} t' & \text{if } \exists\, j \in I \mid minclk(t \Rightarrow p_j,\ Y) = t' \\ +\infty & \text{otherwise} \end{cases} \\
minclk(t \Rightarrow \text{rec } x.p,\ Y) &= minclk(t \Rightarrow p[\text{rec } x.p/x],\ Y) \\
minclk(d_1 \mid d_2,\ Y) &= min\{minclk(d_1,\ Y), minclk(d_2,\ Y)\} \\
minclk(d \setminus B,\ Y) &= minclk(d,\ X), X = \{a \in Y \mid a, \bar{a} \notin B\} \\
minclk(d[\Phi],\ Y) &= minclk(d,\ X), X = \{a \in Act \mid \Phi(a) \in Y\}.
\end{aligned}
$$

The only rule worthy of note is the one regarding summation. Since we are dealing with choices at the same time, we have that either $minclk(t \Rightarrow p_j,\ Y) = \infty$ for every $j \in I$, or, whenever $j, k \in I$ such that $minclk(t \Rightarrow p_j,\ Y) = t'$ and $minclk(t \Rightarrow p_k,\ Y) = t''$, we have $t' = t''$. Note that if d cannot perform any action in Y (for example, when $Y = \varnothing$), we have $minclk(d,\ Y) = +\infty$.

A similar definition can be given to detect the *maximum* active local clock in a state. This function, denoted *maxclk*, is given by replacing *minclk* with *maxclk*, *min* with *max* and $+\infty$ with $-\infty$ in Definition B.1.

$minclk(d,\ Y)$ and $maxclk(d,\ Y)$ are introduced to give a new relation over the set of states, called *over-timing*, which, given two states $d_1, d_2 \in \mathscr{D}_{crav}$, says when d_1 is over-timing

https://doi.org/10.1017/S0960129503004018 Published online by Cambridge University Press

Table 5. *Over-Timing Relation.*

$minclock(d_1)$	$maxclock(d_2)$	$d_1 \rhd\!\!\gg d_2$
$+\infty$	$-\infty$	yes
$+\infty$	n_2	yes
n_1	$-\infty$	yes
n_1	n_2	if and only if $n_1 > n_2$

d_2, that is, when every active local clock in d_1 is strictly greater than any active local clock in d_2.

Definition B.2. Let d_1 and d_2 be $\mathscr{D}_{crav}$ states. d_1 is *over-timing* d_2 (denoted $d_1 \rhd\!\!\gg d_2$) if and only if $minclk(d_1, Act) > maxclk(d_2, Act)$.

Table 5 also says when two $\mathscr{D}_{crav}$ states are in the over-timing relation. Note that every pair of states d_1 and d_2 such that at least one cannot perform any action are in the relation.

Over-timing is not a symmetric relation: states $d_1 = (3 \Rightarrow a + b) \,|\, (5 \Rightarrow nil)$ and $d_2 = (4 \Rightarrow b)$ are such that $d_2 \rhd\!\!\gg d_1$ and $d_1 \not\rhd\!\!\gg d_2$. Regarding $d_3 = (2 \Rightarrow nil)$, we have $d_1 \rhd\!\!\gg d_3$, $d_2 \rhd\!\!\gg d_3$ and $d_3 \rhd\!\!\gg d_1$, $d_3 \rhd\!\!\gg d_2$.

Remark B.1. The statement $maxclk(d, Act) \geqslant minclk(d, Act)$ is false in general. Consider $d = (0 \Rightarrow nil)$ and note that $minclk(d, Act) = +\infty > -\infty = maxclk(d, Act)$.

B.2. *Transitions and the over-timing relation*

In this section we relate over-timing between states and their transitions.

When $minclk(d, Y) = t$, we have that d can perform a transition at starting time t with null execution delay. Of course, a similar statement also holds when $maxclk(d, Y) = t$, as the following lemma shows.

Lemma B.1. Let $d \in \mathscr{D}_{crav}$, $t \in \mathbb{N}$ and $Y \subseteq Act$. $minclk(d, Y) = t$ ($maxclk(d, Y) = t$, respectively) implies

$\exists a \in Y$ and $\exists d' \in \mathscr{D}_{crav}$ such that $d \xrightarrow{\langle a, t+f(a), 0\rangle} d'$.

Proof. The proof is by induction on the depth of the proof of $minclk(d, Y) = t$ (respectively, $maxclk(d, Y) = t$). □

The *minclock* function applied to a timed state $t \Rightarrow p$ is greater than or equal than t. To prove this statement, we focus on open terms as in Lemma A.1.

Lemma B.2. Let $q \in \mathscr{Q}_{crav}$ be such that every free variable is guarded. Then, for every set $\{x_1, \ldots, x_k\} \supseteq \mathscr{F}(q)$, $\forall p_{x_1}, \ldots, p_{x_k} \in \mathscr{P}$, $\forall\, t \in \mathbb{N}$ and $\forall\, Y \subseteq Act$, we have

$$minclk(t \Rightarrow q[p_{x_1}/x_1, \ldots, p_{x_k}/x_k], Y) \geqslant t.$$

https://doi.org/10.1017/S0960129503004018 Published online by Cambridge University Press

Proof. The proof proceeds by structural induction on q and is similar in all respects to that of Lemma A.1. Again we will only prove the more involved case when q is a recursive process. The others are simpler and follow by simple inductive reasoning.

Assume $q = \text{rec } x.q_1$. As in Lemma A.1, we can assume that the variables in q are in $\{x_1,\ldots,x_k\}$ and that $x \notin \{x_1,\ldots,x_k\}$. By the hypothesis, x is free and guarded in q_1. Hence, $\{x_1,\ldots,x_k,x\} \supseteq \mathscr{F}(q_1)$. By the induction hypothesis,

$$minclk\,((t \Rightarrow q_1[p_{x_1}/x_1,\ldots,p_{x_k}/x_k,p/x]),\ Y) \geqslant t, \qquad \forall p \in \mathscr{P}.$$

The state $t \Rightarrow q[p_{x_1}/x_1,\ldots,p_{x_k}/x_k]$ coincides with

$$d = t \Rightarrow \text{rec } x.(q_1[p_{x_1}/x_1,\ldots,p_{x_k}/x_k]).$$

Thus,

$$\begin{aligned} &minclk\,(d,\ Y)\\ &= minclk\,(t \Rightarrow \text{rec } x.(q_1[p_{x_1}/x_1,\ldots,p_{x_k}/x_k]),\ Y)\\ &= minclk\,(t \Rightarrow (q_1[p_{x_1}/x_1,\ldots,p_{x_k}/x_k])[\text{rec } x.(q_1[p_{x_1}/x_1,\ldots,p_{x_k}/x_k])/x],\ Y)\\ &= minclk\,(t \Rightarrow q_1[p_{x_1}/x_1,\ldots,p_{x_k}/x_k,\text{rec } x.(q_1[p_{x_1}/x_1,\ldots,p_{x_k}/x_k])/x],\ Y). \end{aligned}$$

By the induction hypothesis (let $p = \text{rec } x.q_1[p_{x_1}/x_1,\ldots,p_{x_k}/x_k]$), $minclk\,(d,\ Y) \geqslant t$. □

As a corollary to the previous lemma, we have a similar result in the class of closed $\mathscr{D}_{crav}$ states (and, hence, $\mathscr{P}_{crav}$ processes also).

Proposition B.1. Let $t \in \mathbb{N}$, $Y \subseteq Act$. Then:

(i) $p \in \mathscr{P}_{crav}$ implies $minclk\,(t \Rightarrow p,\ Y) \geqslant t$.

(ii) $p \in \mathscr{S}$, $a \in Y$ and $(t \Rightarrow p) \xrightarrow{\langle a,t',0\rangle} d$ imply $minclk\,(t \Rightarrow p,\ Y) = t$.

Proof. Item (i) follows by Lemma B.2. Just note that since p is a process, we have $\mathscr{F}(p) = \varnothing$. Item (ii) follows by induction on the depth of the proof of $(t \Rightarrow p) \xrightarrow{\langle a,t',0\rangle} d$. Hypothesis $p \in \mathscr{S}$ is essential. As a counterexample, consider $p = (3)a.nil (\notin \mathscr{S})$, and note that $minclk\,(t \Rightarrow (3)a.nil,\ Y) = t + 3 (> t)$. □

The next lemma states an invariant property of process transitions regarding the minimum active clock.

Proposition B.2. Let $d_1, d_2 \in \mathscr{D}_{crav}$, $t \in \mathbb{N}$, $t^\star \in \mathbb{N}^\star$, $Y \subseteq Act$ and $a \in Y$ be such that $d_1 \xrightarrow{\langle a,t,0\rangle} d_2$. Then, $minclk\,(d_1,\ Y) > t^\star$ implies $minclk\,(d_2,\ Y) > t^\star$.

Proof. Consider $d_1, d_2 \in \mathscr{D}_{crav}$, $t \in \mathbb{N}$, $t^\star \in \mathbb{N}^\star$, $Y \subseteq Act$, $a \in Y$, $d_1 \xrightarrow{\langle a,t,0\rangle} d_2$ and $minclk\,(d_1,\ Y) > t^\star$. The proof is by induction on the depth of the proof of derivation $d_1 \xrightarrow{\langle a,t,0\rangle} d_2$. We proceed by case analysis on the structure of d_1. When $d_1 = t' \Rightarrow p$ for $t' \in \mathbb{N}$ and $p \in \mathscr{P}$, we assume $t' = t - f(a)$.

(a) $d_1 = (t' \Rightarrow nil)$. This case is impossible.

(b) $d_1 = (t' \Rightarrow b.p)$. If $b \neq a$, the case is impossible. Let $d_1 = (t' \Rightarrow a.p)$ and $d_2 = (t' + f(a) \Rightarrow p)$. By the definition of *minclock*, we have $minclk\,(d_1,\ Y) = t'$ and, by the

https://doi.org/10.1017/S0960129503004018 Published online by Cambridge University Press

hypothesis $minclk(d_1, Y) > t^\star$, we have $t' > t^\star$. By Proposition B.1, it follows that $minclk(d_2, Y) \geqslant t' + f(a)$. Hence, we have $minclk(d_2, Y) \geqslant t' + f(a) > t^\star + f(a) > t^\star$. Thus, $minclk(d_2, Y) > t^\star$.

(c) $d_1 = t' \Rightarrow (n)p$. Then $t' \Rightarrow (n)p \xrightarrow{\langle a,t,0\rangle} d_2$ if $t' + n \Rightarrow p \xrightarrow{\langle a,t,0\rangle} d_2$.
Moreover, $minclk(t' \Rightarrow (n)p, Y) > t^\star$ implies $minclk(t' + n \Rightarrow p, Y) > t^\star$. By the induction hypothesis, $minclk(d_2, Y) > t^\star$.

(d) $d_1 = (t' \Rightarrow \sum_{i\in i} p_i)$. By the operational rules, transition $d_1 \xrightarrow{\langle a,t,0\rangle} d_2$ is derivable if and only if $j \in I$ such that $(t' \Rightarrow p_j) \xrightarrow{\langle a,t,0\rangle} d_2$. Transition $(t' \Rightarrow p_j) \xrightarrow{\langle a,t,0\rangle} d_2$ and Lemma B.1(ii) imply $minclk(t' \Rightarrow p_j, Y) = t'$. Thus $minclk(t' \Rightarrow \sum_{i\in i} p_i, Y) = t'$, since $\exists j \in I$ such that $minclk(t' \Rightarrow p_j, Y) = t'$. Moreover, $t' > t^\star$. Therefore, by $minclk(t' \Rightarrow p_j, Y) > t^\star$ and transition $(t' \Rightarrow p_j) \xrightarrow{\langle a,t,0\rangle} d_2$, we can conclude that $minclk(d_2, Y) > t^\star$.

(e) $d_1 = d_3 \,|\, d_4$. Then $d_1 \xrightarrow{\langle a,t,0\rangle} d_2$ is derivable if $d_3 \xrightarrow{\langle a,t,0\rangle} d_3'$ and $d_2 = d_3' \,|\, d_4$ (the symmetric case is similar). By the definition of *minclock*, by $minclk(d_3 \,|\, d_4, Y) > t^\star$, we have $minclk(d_3, Y) > t^\star$ and $minclk(d_4, Y) > t^\star$. Then, by the induction hypothesis, $d_3 \xrightarrow{\langle a,t,0\rangle} d_3'$ and $minclk(d_3, Y) > t^\star$ gives $minclk(d_3', Y) > t^\star$. Then $minclk(d_2, Y) = minclk(d_3' \,|\, d_4, Y) > t^\star$.

(f) Cases $d_1 = d \setminus B$, $d_1 = d[\Phi]$ and $d_1 = (t' \Rightarrow \mathrm{rec}\ x.p)$ follow by simpler inductive reasoning. □

Remark B.2. Condition $a \in Y$ in Proposition B.2 is strictly needed. For $a \in Act(1)$, $t^\star = 4$ and $d_1 = (3 \Rightarrow a.b \,|\, 5 \Rightarrow b) \xrightarrow{\langle a,4,0\rangle} d_2 = (4 \Rightarrow b \,|\, 5 \Rightarrow b)$, we have $minclk(d_1, \{b\}) = 5 > t^\star = minclk(d_2, \{b\})$.

The following corollary generalises Proposition B.2.

Corollary B.1. Let $d_1, d_2, d_3 \in \mathscr{D}_{crav}$, $t \in \mathbb{N}$ and $a \in Act$ be such that $d_1 \xrightarrow{\langle a,t,0\rangle} d_2$. Then, $d_1 \rhd\!\!\gg d_3$ implies $d_2 \rhd\!\!\gg d_3$.

Proof. Assume $d_1, d_2, d_3 \in \mathscr{D}_{crav}$, $t \in \mathbb{N}$, $a \in Act$ and $d_1 \rhd\!\!\gg d_3$. Moreover, assume $d_1 \xrightarrow{\langle a,t,0\rangle} d_2$. From the definition of over-timing, $d_1 \rhd\!\!\gg d_3$ implies $minclk(d_1, Act) > maxclk(d_3, Act)$. By fixing $t^\star = maxclk(d_3, Act)$, transition $d_1 \xrightarrow{\langle a,t,0\rangle} d_2$ and Proposition B.2 imply $minclk(d_2, Act) > t^\star$. Thus, $minclk(d_2, Act) > maxclk(d_3, Act)$ implies $d_2 \rhd\!\!\gg d_3$. □

The following lemma states that the starting time of a transition is between the minimum and maximum active local clocks of the source state.

Lemma B.3. Let $d, d' \in \mathscr{D}_{crav}$, $Y \subseteq Act$, $a \in Y$ and $t, t^\star \in \mathbb{N}$. If $d \xrightarrow{\langle a,t,0\rangle} d'$ and $t^\star = t - f(a)$, then $maxclk(d, Y) \geqslant t^\star \geqslant minclk(d, Y)$.

Proof. Assume $d, d' \in \mathscr{D}_{crav}$, $Y \subseteq Act$, $a \in Y$, $t, t^\star \in \mathbb{N}$ such that $d \xrightarrow{\langle a,t,0\rangle} d'$ and $t^\star = t - f(a)$. By induction on the depth of the transition $d \xrightarrow{\langle a,t,0\rangle} d'$. We proceed by case analysis on the syntactic structure of d.

(a) $d = t' \Rightarrow nil$ is impossible.

https://doi.org/10.1017/S0960129503004018 Published online by Cambridge University Press

(b) $d = t' \Rightarrow b.p$ is possible if and only if $b = a$. Thus, $d = t' \Rightarrow a.p$ and $d' = t' + f(a) \Rightarrow p$. Then $minclk(t' \Rightarrow a.p,\ Y) = maxclk(t' \Rightarrow a.p,\ Y) = t'$.

(c) $d = t' \Rightarrow (n)p$. Transition $d \xrightarrow{\langle a,t,0\rangle} d'$ is derivable if $t' + n \Rightarrow p \xrightarrow{\langle a,t,0\rangle} d'$. By the induction hypothesis on the latter transition, we have $maxclk(t' + n \Rightarrow p,\ Y) \geqslant t^\star \geqslant minclk(t' + n \Rightarrow p,\ Y)$. Then note that $minclk(t' + n \Rightarrow p,\ Y) = minclk(d,\ Y)$ and $maxclk(t' + n \Rightarrow p,\ Y) = maxclk(d,\ Y)$.

(d) $d = d_1 \mid d_2$. From the operational rules, we have the transition $d \xrightarrow{\langle a,t,0\rangle} d'$ is derivable if $d_1 \xrightarrow{\langle a,t,0\rangle} d_1'$ and $d' = d_1' \mid d_2$ (the symmetric case is similar). By the induction hypothesis, we have $maxclk(d_1,\ Y) \geqslant t^\star \geqslant minclk(d_1,\ Y)$.
The statement then follows by $minclk(d_1,\ Y) \geqslant minclk(d,\ Y)$ and $maxclk(d_1,\ Y) \leqslant maxclk(d,\ Y)$.

(e) $d = d_1 \backslash B$. From the operational rules, we have the transition $d \xrightarrow{\langle a,t,0\rangle} d'$ is derivable if $d_1 \xrightarrow{\langle a,t,0\rangle} d_1'$ with $a, \bar{a} \notin B$ and $d' = d_1' \backslash B$. By the induction hypothesis, by letting $X = \{c \in Y \mid c, \bar{c} \notin B\}$, we have $maxclk(d_1,\ X) \geqslant t^\star \geqslant minclk(d_1,\ X)$. The statement follows by noting that $minclk(d_1,\ X) = minclk(d,\ Y)$ and $maxclk(d_1,\ X) = maxclk(d,\ Y)$.

(f) Cases $d = t' \Rightarrow \sum_{i \in I} p_i$, $d = t' \Rightarrow \mathrm{rec}\ x.p$ and $d = d_1[\Phi]$ follow as in the previous cases.

□

Proposition B.3. Let $a \in Act$, $t_i \in \mathbb{N}$, $d_i, d_i' \in \mathscr{D}_{crav}$, $d_1 \rhd\!\!\rhd d_2$ and $d_i \xrightarrow{\langle a,t_i,0\rangle} d_i'$ for each $i \in \{1,2\}$. Then, $t_1 > t_2$.

Proof. Let $a \in Act$, $t_i, t_i' \in \mathbb{N}$ and $d_i, d_i' \in \mathscr{D}_{crav}$ for each $i \in \{1,2\}$. By the over-timing hypothesis, $d_1 \rhd\!\!\rhd d_2$, we have $minclk(d_1,\ Act) > maxclk(d_2,\ Act)$. Lemma B.3 applied to transitions $d_1 \xrightarrow{\langle a,t_1,0\rangle} d_1'$ and $d_2 \xrightarrow{\langle a,t_2,0\rangle} d_2'$ gives $maxclk(d_1,\ Act) \geqslant t_1 - f(a) \geqslant minclk(d_1,\ Act)$ and $maxclk(d_2,\ Act) \geqslant t_2 - f(a) \geqslant minclk(d_2,\ Act)$. By $minclk(d_1,\ Act) > maxclk(d_2,\ Act)$, we have $t_1 - f(a) > t_2 - f(a)$, from which we can conclude $t_1 > t_2$. □

B.3. *Only parallel contexts*

An *only parallel context*, denoted $O_{[]}$, is a term with a single 'hole' of the grammar

$$O_{[]} ::= [] \ \Big|\ O_{[]} \mid d \ \Big|\ d \mid O_{[]},$$

where d is a state.

Transitions with null execution delay out of states d and $O[d]$ are strictly related.

Lemma B.4. Let $d_1, d_2 \in \mathscr{D}_{crav}$, $a \in Act$, $t \in \mathbb{N}$ and $O_{[]}$ be an only parallel context. Then, $d_1 \xrightarrow{\langle a,t,0\rangle} d_2$ if and only if $O[d_1] \xrightarrow{\langle a,t,0\rangle} O[d_2]$.

Proof. Assume $d_1 \in \mathscr{D}_{crav}$, $a \in Act$, $t \in \mathbb{N}$ and $O_{[]}$ be an only parallel context. We proceed by induction on the syntactical structure of $O_{[]}$.

(a) $O_{[]} = []$. This case is simple because $O[d_j] = d_j$, for every $j \in \{1,2\}$.

https://doi.org/10.1017/S0960129503004018 Published online by Cambridge University Press

(b) $O_{[]} = d \mid O'_{[]}$, $d \in \mathscr{D}_{crav}$. In this case we have that $O[d_1] \xrightarrow{\langle a,t,0\rangle} O[d_2]$ if and only if $d \mid O'[d_1] \xrightarrow{\langle a,t,0\rangle} d \mid O'[d_2]$ if and only if $O'[d_1] \xrightarrow{\langle a,t,0\rangle} O'[d_2]$, and, by the induction hypothesis, if and only if $d_1 \xrightarrow{\langle a,t,0\rangle} d_2$.

(c) Case $O_{[]} = O'_{[]} \mid d$, with $d \in \mathscr{D}_{crav}$, is similar to the previous one. □

The above lemma cannot be extended to deal with parallel contexts (see Section B) instead of only parallel ones. In the former case, indeed, the occurrence of [] may appear within the scope of a restriction or relabelling operator. This might prevent some actions out of d from happening in $C[d]$.

The following Lemma states an intuitive result. When two states are eager equivalent they necessarily have the same minimum active local clock.

Lemma B.5. Let $d_1, d_2 \in \mathscr{D}_{crav}$. If $d_1 \sim_e d_2$, then $minclk(d_1, Act) = minclk(d_2, Act)$.

Proof. Assume, for a contradiction, two states $d_1, d_2 \in \mathscr{D}_{crav}$ such that $d_1 \sim_e d_2$ and $minclk(d_1, Act) > minclk(d_2, Act)$ (case $minclk(d_1, Act) < minclk(d_2, Act)$ follows by similar reasoning). The case $minclk(d_1, Act) = +\infty$ and $minclk(d_2, Act) \in \mathbb{N}$ is impossible, since otherwise $d_1 \not\sim_e d_2$. By Lemma B.1, a transition $d_2 \xrightarrow{\langle a,t,0\rangle} d'_2$ with $t = minclk(d_2, Act) + f(a)$ is derivable. By $d_1 \sim_e d_2$, we also have $d_1 \xrightarrow{\langle a,t,0\rangle} d'_1$. By hypothesis $t - f(a) < minclk(d_1, Act)$. This contradicts Lemma B.3. □

As in the previous section, we need a new bisimulation equivalence called (t)-*Eager Equivalence* and denoted $\sim_e^{(t)}$. It plays a similar role to $\sim_t^n$. (t)-*Eager Equivalence* concentrates on visible actions and relates processes able to perform the same actions at the same starting time t.

Definition B.3 ((t)-Eager Equivalence).

(1) Let $t \in \mathbb{N}$. The functional $E^{(t)} : Rel \to Rel$ is defined, for each $\mathfrak{R} \in Rel$, as follows: $(d_1, d_2) \in E^{(t)}(\mathfrak{R})$ if, for each $a \in Act$,
— $d_1 \xrightarrow{\langle a,t+f(a),0\rangle} d'_1$ implies $d_2 \xrightarrow{\langle a,t+f(a),0\rangle} d'_2$ for some $d'_2 \in \mathscr{D}_{crav}$ such that $(d'_1, d'_2) \in \mathfrak{R}$.

(2) A relation $\mathfrak{R} \in Rel$ will be called $E^{(t)}$-*bisimulation* if $\mathfrak{R} \subseteq E^{(t)}(\mathfrak{R})$.

(3) We say that two states d_1 and d_2 are (t)-*eager equivalent*, $d_1 \sim_e^{(t)} d_2$, if and only if there exists an $E^{(t)}$-bisimulation $\mathfrak{R}$ such that $(d_1, d_2) \in \mathfrak{R}$.

(4) We say that two processes p_1 and p_2 are (t)-*eager equivalent*, $p_1 \sim_e^{(t)} p_2$, if and only if $(0 \Rightarrow p_1) \sim_e^{(t)} (0 \Rightarrow p_2)$.

Again, eager equivalence is strictly finer than (t)-eager equivalence and $\sim_e^{(t)}$ is preserved by all operators of the language.

Proposition B.4. Let $p_1, p_2 \in \mathscr{P}_{crav}$ be such that $p_1 \sim_e p_2$. Then, $p_1 \sim_e^{(t)} p_2$ for each $t \in \mathbb{N}$.

For the next lemma we assume $minclk([], Act) = \infty$ and $maxclk([], Act) = -\infty$.

Lemma B.6. Let $d \in \mathscr{D}_{crav}$, $t \in \mathbb{N}$ and O be an only parallel context such that $d \rhd\!\!\rhd O$ and $t > maxclk(O, Act)$. Then, $O[d] \sim_e^{(t)} d$.

https://doi.org/10.1017/S0960129503004018 Published online by Cambridge University Press

Proof. By Lemma B.4, $d \xrightarrow{\langle a,t',0\rangle} d'$ if and only if $O[d] \xrightarrow{\langle a,t',0\rangle} O[d']$. Moreover, by $t > maxclk(O, Act)$, there is no $a \in Act$ and $O'[]$ an only parallel context such that $O[] \xrightarrow{\langle a,t+f(a),0\rangle} O'[]$. Then it is easy to prove that the relation (Corollary B.1 is also needed)

$$\mathfrak{R}^t = \{(O[d], d) \mid d \in \mathscr{D}_{crav},\ O \text{ a only parallel context}, t > maxclk(O, Act) \text{ and } d \rhd\!\gg O\}$$

is an $E^{(t)}$-*bisimulation.* □

B.4. *The main results*

The following decomposition lemma is the key statement to prove our main result. Consider two only parallel contexts $O'_{[]}, O''_{[]}$ and $d', d'' \in \mathscr{D}_{crav}$. Assume $O'[d'] \sim_e O''[d'']$. We state conditions under which $O'[d'] \sim_e O''[d'']$ implies the eager equivalence of d', d'', and $O'_{[]}, O''_{[]}$ in isolation. More precisely, we prove that when each active local clock within d' and d'' is strictly greater than each active local clock within $O'_{[]}$ and $O''_{[]}$, we have $O'[d'] \sim_e O''[d'']$ implies $d' \sim_e d''$ and $O' \sim_e O''$.

Lemma B.7. Let $d', d'' \in \mathscr{D}_{crav}$ and $O'_{[]}, O''_{[]}$ be two only parallel contexts such that the following hold:

(1) $d \rhd\!\gg O$, with $d \in \{d', d''\}$ and $O \in \{O', O''\}$;
(2) $O'[d'] \sim_e O''[d'']$.

Then, $d' \sim_e d''$ and $O' \sim_e O''$.

Proof. Assume that (1) and (2) hold for $d', d'' \in \mathscr{D}_{crav}$ and $O'_{[]}, O''_{[]}$. In the rest of this proof we will often write d_n and $O_{m[]}$ (with $n, m \in \{1, 2\}$) to denote a pair of states d_1, d_2 and a pair of only parallel contexts $O_{1[]}, O_{2[]}$, respectively. We prove the two statements separately.

(a) *Proof of* $d' \sim_e d''$. Consider

$$\mathfrak{R}_1 = \{(d_1, d_2) \mid \exists\, O_{1[]}, O_{2[]} \text{ such that } O_1[d_1] \sim_e O_2[d_2] \text{ and } d_n \rhd\!\gg O_m\}.$$

We prove that $\mathfrak{R}_1$ is an $\sim_e$-bisimulation.
Clearly, $(d', d'') \in \mathfrak{R}_1$. So, we can assume $(d_1, d_2) \in \mathfrak{R}_1$ with $O_{1[]}$ and $O_{2[]}$ such that $O_1[d_1] \sim_e O_2[d_2]$ and $d_n \rhd\!\gg O_m$. Moreover, assume $d_1 \xrightarrow{\langle a,t,0\rangle} d'_1$ (the symmetric case is similar). From Lemma B.4 applied to the latter transition, $O_1[d_1] \xrightarrow{\langle a,t,0\rangle} O_1[d'_1]$ is derivable. From the hypothesis $O_1[d_1] \sim_e O_2[d_2]$, we also have $O_2[d_2] \xrightarrow{\langle a,t,0\rangle} d''_2$ such that $O_1[d'_1] \sim_e d''_2$. From $d_1 \xrightarrow{\langle a,t,0\rangle} d'_1$, Proposition B.3 and the fact that $d_1 \rhd\!\gg O_2$, we must have $O_2[d_2] \xrightarrow{\langle a,t,0\rangle} O[d'_2]$ for some d'_2 such that $d_2 \xrightarrow{\langle a,t,0\rangle} d'_2$ (or, in other words, transitions out of O_2 performing an a action are only possible at completion time strictly less than t – hence $O_2 \xrightarrow{\langle a,t,0\rangle} O'_2$ is impossible). Transitions $d_n \xrightarrow{\langle a,t,0\rangle} d'_n$, the hypothesis $d_n \rhd\!\gg O_m$ and Corollary B.1, allow us to conclude that $d'_n \rhd\!\gg O_m$. Hence, from $O_1[d'_1] \sim_e O_2[d'_2]$, we have $(d'_1, d'_2) \in \mathfrak{R}_1$.

https://doi.org/10.1017/S0960129503004018 Published online by Cambridge University Press

(b) *Proof of* $O' \sim_e O''$. Consider

$$\mathfrak{R}_2 = \{(O_1, O_2) \mid \exists\, d_1, d_2 \in \mathscr{D}_{crav} \text{ such that } O_1[d_1] \sim_e O_2[d_2] \text{ and } d_n \rhd\!\!\rhd O_m\}.$$

We prove that $\mathfrak{R}_2$ is an $\sim_e$-bisimulation. Clearly, $(O', O'') \in \mathfrak{R}_2$. Assume $(O_1, O_2) \in \mathfrak{R}_2$. Then, $d_1, d_2 \in \mathscr{D}_{crav}$ such that $O_1[d_1] \sim_e O_2[d_2]$ and $d_n \rhd\!\!\rhd O_m$. Moreover, assume $O_1 \xrightarrow{\langle a,t,0\rangle} O_1'$ (the symmetric case is similar) for some $a \in Act$ and $t \in \mathbb{N}$. We consider two steps.

(b_1) We show that there are two states d_n', derivatives of d_n, which are such that $O_1[d_1'] \sim_e O_2[d_2']$, $d_n' \rhd\!\!\rhd O_m$ and $minclk\,(d_n', Act\,) > maxclk\,(O_1', Act\,)$. We will find d_n' by performing all possible transitions out of d_n at starting time from 0 to $maxclk\,(O_1', Act\,)$. Since each process in the current language can only perform a finite number of concurrent actions at a given starting time (see Proposition A.1), this procedure is actually effective.

First we prove for every $i \geqslant 0$ that there are d_n' derivatives of d_n such that $O_1[d_1'] \sim_e O_2[d_2']$, $d_n' \rhd\!\!\rhd O_m$ and $minclk\,(d_n', Act\,) > i$. We proceed by induction on $maxclk\,(O_1', Act\,)$ (which we assume to be greater than $-\infty$, otherwise the proof is trivial).

The case $i = 0$ is simple as $O_1[d_1] \sim_e O_2[d_2]$, $d_n \rhd\!\!\rhd O_m$ and $minclk\,(d_n, Act\,) \geqslant 0$ (each local clock within a state is greater than or equal to 0). Now, we assume the statement for $i-1$ and prove it for i. Hence there are states $d_n^{(k_i)}$, which are derivatives of d_n such that $O_1[d_1^{(k_i)}] \sim_e O_2[d_2^{(k_i)}]$, $d_n^{(k_i)} \rhd\!\!\rhd O_m$, $minclk\,(d_n^{(k_i)}, Act\,) > i-1$, and prove that there exist $d_n^{(k_{i+1})}$ derivatives of $d_n^{(k_i)}$ such that $O_1[d_1^{(k_{i+1})}] \sim_e O_2[d_2^{(k_{i+1})}]$, $d_n^{(k_{i+1})} \rhd\!\!\rhd O_m$ and $minclk\,(d_n^{(k_{i+1})}, Act\,) > i$.

Consider the following computation at starting time $i - \mathscr{C}_1$:

$$d_1^{(k_i)} \xrightarrow{\langle a_{k_i}, t_{k_i}, 0\rangle} d_1^{(k_i+1)} \xrightarrow{\langle a_{k_i+1}, t_{k_i+1}, 0\rangle} \cdots \xrightarrow{\langle a_{k_{i+1}-1}, t_{k_{i+1}-1}, 0\rangle} d_1^{(k_{i+1})},$$

where $k_{i+1} \geqslant k_i$ and $i = t_j - f(a_j)$ for every $k_i \leqslant j < k_{i+1}$. Suppose that state $d_1^{(k_{i+1})}$ cannot perform eager actions at starting time i. Computation $\mathscr{C}_1$ must be finite, that is, k_{i+1} exists by Proposition A.1 (otherwise, state $d_1^{(k_i)}$ would have an infinite computation at starting time i). From $d_1^{(k_i)} \rhd\!\!\rhd O_m$, Corollary B.1 applies repeatedly to each transition within $\mathscr{C}_1$. Thus we have $d_1^{(j)} \rhd\!\!\rhd O_m$, $\forall j$ such that $k_i \leqslant j \leqslant k_{i+1}$. Moreover, by the hypothesis, $minclk\,(d_1^{(k_i)}, Act\,) > i-1$. Thus, by Proposition B.2, we have $minclk\,(d_1^{(j)}, Act\,) > i-1$, $\forall j$ such that $k_i \leqslant j \leqslant k_{i+1}$. In particular, we have $minclk\,(d_1^{(k_{i+1})}, Act\,) > i-1$. From Lemma B.1 we have $minclk\,(d_1^{(k_{i+1})}, Act\,) \neq i$ (otherwise we would have an eager transition out of $d_1^{(k_{i+1})}$ at starting time i and this would contradict the hypothesis). Hence, $minclk\,(d_1^{(k_{i+1})}, Act\,) > i$.

By Lemma B.4 we have the following computation – $\mathscr{C}_1'$:

$$O_1[d_1^{(k_i)}] \xrightarrow{\langle a_{k_i}, t_{k_i}, 0\rangle} O_1[d_1^{(k_i+1)}] \xrightarrow{\langle a_{k_i+1}, t_{k_i+1}, 0\rangle} \cdots \xrightarrow{\langle a_{k_{i+1}-1}, t_{k_{i+1}-1}, 0\rangle} O_1[d_1^{(k_{i+1})}].$$

From the hypothesis $O_1[d_1^{(k_i)}] \sim_e O_2[d_2^{(k_i)}]$, we also have a computation – $\mathscr{C}_2'$:

$$O_2[d_2^{(k_i)}] \xrightarrow{\langle a_{k_i}, t_{k_i}, 0\rangle} O_2[d_2^{(k_i+1)}] \xrightarrow{\langle a_{k_i+1}, t_{k_i+1}, 0\rangle} \cdots \xrightarrow{\langle a_{k_{i+1}-1}, t_{k_{i+1}-1}, 0\rangle} O_2[d_2^{(k_{i+1})}],$$

https://doi.org/10.1017/S0960129503004018 Published online by Cambridge University Press

with $O_1[d_1^{(j)}] \sim_e O_2[d_2^{(j)}]$, $\forall j$ such that $k_i \leqslant j \leqslant k_{i+1}$ can be derived. Indeed, from $d_1^{(j)} \rhd\!\!\rhd O_2$ ($\forall j$ such that $k_i \leqslant j \leqslant k_{i+1}$) and $d_1^{(k_i)} \xrightarrow{\langle a_{k_i}, t_{k_i}, 0\rangle} d_1^{(k_i+1)}$, we have that each eager transition out of O_2 is derivable at a completion time strictly less than t_{k_i} (see Proposition B.3). In other words, in computation $\mathscr{C}'_2$, context $O_{2[]}$ is never involved. Moreover, the state $O_2[d_2^{(k_{i+1})}]$ cannot perform eager actions at starting time i. Otherwise, from $O_1[d_1^{(k_{i+1})}] \sim_e O_2[d_2^{(k_{i+1})}]$, state $d_1^{(k_{i+1})}$ would also be able to match this transition thereby contradicting the hypothesis.
From Lemma B.4 we also have – $\mathscr{C}_2$

$$d_2^{(k_i)} \xrightarrow{\langle a_{k_i}, t_{k_i}, 0\rangle} d_2^{(k_i+1)} \xrightarrow{\langle a_{k_i+1}, t_{k_i+1}, 0\rangle} \cdots \xrightarrow{\langle a_{k_{i+1}-1}, t_{k_{i+1}-1}, 0\rangle} d_2^{(k_{i+1})}.$$

As in the case of computation $\mathscr{C}_1$, we have that Corollary B.1, Proposition B.2 and Lemma B.1 (note that $d_2^{(k_i)} \rhd\!\!\rhd O_m$ and $minclk\,(d_2^{(k_i)}, Act) > i-1$) allow us to prove that $d_2^{(j)} \rhd\!\!\rhd O_m$, $minclk\,(d_2^{(j)}, Act) > i-1$ ($\forall j$ such that $k_i \leqslant j \leqslant k_{i+1}$) and $minclk\,(d_2^{(k_{i+1})}, Act) > i$.
Thus, we can conclude that there are two states $d_n^{(k_{i+1})}$ (which are derivatives of $d_n^{(k_i)}$) such that we have $O_1[d_1^{(k_{i+1})}] \sim_e O_2[d_2^{(k_{i+1})}]$, $d_n^{(k_{i+1})} \rhd\!\!\rhd O_m$ and $minclk\,(d_n^{(k_{i+1})}, Act) > i$.

(b_2) Assume two states d'_1 and d'_2 (obtained from step (b_1) by choosing $d_1^{(k_{s+1})}$ and $d_2^{(k_{s+1})}$, respectively, with $s = maxclk\,(O'_1, Act)$) such that

(i) $O_1[d'_1] \sim_e O_2[d'_2]$,

(ii) $d'_n \rhd\!\!\rhd O_m$ and

(iii) $minclk\,(d'_n, Act) > maxclk\,(O'_1, Act)$.

From $O_1 \xrightarrow{\langle a,t,0\rangle} O'_1$, we have that $O_1[d'_1] \xrightarrow{\langle a,t,0\rangle} O'_1[d'_1]$, and from (i), we have $O_2[d'_2] \xrightarrow{\langle a,t,0\rangle} d''_2$ such that $O'_1[d'_1] \sim_e d''_2$. By considering $O_1 \xrightarrow{\langle a,t,0\rangle} O'_1$ and the overtiming hypothesis $d'_2 \rhd\!\!\rhd O_1$ of item (ii), by Proposition B.3, we have $d''_2 = O'_2[d'_2]$ and $O_2 \xrightarrow{\langle a,t,0\rangle} O'_2$. We have $O_m \xrightarrow{\langle a,t,0\rangle} O'_m$, and there are two states d'_n such that $O'_1[d'_1] \sim_e O'_2[d'_2]$. In order to state that $(O'_1, O'_2) \in \mathfrak{R}_2$, it remains to prove that $d'_n \rhd\!\!\rhd O'_m$. From the hypothesis we know $minclk\,(d'_n, Act) > maxclk\,(O'_1, Act)$. Clearly, $d'_n \rhd\!\!\rhd O'_1$. Therefore we have to prove that $d'_n \rhd\!\!\rhd O'_2$. We start by proving $d'_2 \rhd\!\!\rhd O'_2$. Assume $minclk\,(d'_2, Act) \leqslant maxclk\,(O'_2, Act)$, for a contradiction. By Lemma B.1, for some only parallel context O''_2, $b \in Act$ and $t', t'' \in \mathbb{N}$, there is a transition $O'_2 \xrightarrow{\langle b,t',0\rangle} O''_2$ with $t'' = t' - f(b) \geqslant minclk\,(d'_2, Act)$. By Item (a), we have $d'_1 \sim_e d'_2$ (see items (i) and (ii)). By Proposition B.4, $d'_1 \sim_e^{(t'')} d'_2$ immediately follows.
By Lemma B.5, $minclk\,(d'_2, Act) = minclk\,(d'_1, Act)$. Moreover, by hypothesis (Item (iii)), $minclk\,(d'_1, Act) > maxclk\,(O'_1, Act)$. Therefore,

$$t'' \geqslant minclk\,(d'_2, Act) = minclk\,(d'_1, Act) > maxclk\,(O'_1, Act).$$

Hence $t'' > maxclk\,(O'_1, Act)$ and $d'_1 \rhd\!\!\rhd O'_1$ (Item (iii)) can be used together with Lemma B.6 to conclude $d'_1 \sim_e^{(t'')} O'_1[d'_1]$. By considering $O'_1[d'_1] \sim_e O'_2[d'_2]$, we have

https://doi.org/10.1017/S0960129503004018 Published online by Cambridge University Press

$O_1'[d_1'] \sim_e^{(t'')} O_2'[d_2']$ by Proposition B.4. Now we prove $O_2'[d_2'] \not\sim_e^{(t'')} d_2'$. This implies, by transitivity of $\sim_e^{(t'')}$, that $d_1' \not\sim_e^{(t'')} d_2'$, thereby contradicting the hypothesis.
Suppose for a contradiction that $O_2'[d_2'] \sim_e^{(t'')} d_2'$. Since we already know that $O_2[d_2'] \sim_e^{(t'')} d_2'$, by transitivity, we have $O_2[d_2'] \sim_e^{(t'')} O_2'[d_2']$. Moreover, from $d_2' \rhd\!\!\gg O_2$, we know that

$$t'' \geqslant minclk(d_2', Act) > maxclk(O_2, Act).$$

By Lemma B.3, O_2 cannot perform actions at starting time t''. Thus, as in case (b_1) above, if we perform all the transitions at starting time t'' that d_2' in $O_2'[d_2']$ can perform, then the state $O_2[d_2']$ must match these transitions ($O_2[d_2'] \sim_e^{(t'')} O_2'[d_2']$) with transitions out of d_2'. Still, as in case (b_1), by Proposition A.1, these two computations must be finite (otherwise, states $O_2[d_2']$ and $O_2'[d_2']$ would have an infinite computation at starting time t''). Assume that the target states of these computations out of $O_2[d_2']$ and $O_2'[d_2']$ are $O_2[d_2'']$ and $O_2'[d_2''']$, respectively. By the hypothesis, the latter state can still perform $O_2' \xrightarrow{\langle b,t',0\rangle} O_2''$ at starting time t''. This transition cannot be matched by $O_2[d_2'']$ because, by the hypothesis, O_2 and d_2'' cannot both perform transitions at starting time t''.
Hence, we have $O_2'[d_2'] \not\sim_e^{(t'')} d_2'$. This means that $d_1' \not\sim_e^{(t'')} d_2'$, and we find a contradiction.
Then we have $minclk(d_2', Act) > maxclk(O_2', Act)$ (that is, $d_2' \rhd\!\!\gg O_2'$).
Finally, by

$$minclk(d_2', Act) = minclk(d_1', Act)$$

and

$$minclk(d_1', Act) > maxclk(O_2', Act)$$

(that is, $d_1' \rhd\!\!\gg O_2'$), we can conclude $d_n' \rhd\!\!\gg O_2'$, and therefore $d_n' \rhd\!\!\gg O_m'$.

Summarising, items (b_1) and (b_2) provide two states d_n' such that $O_1'[d_1'] \sim_e O_2'[d_2']$ and $d_n' \rhd\!\!\gg O_m'$. Hence, $(O_1', O_2') \in \mathfrak{R}_2$. □

As a first application of this decomposition statement, consider the following useful lemma.

Lemma B.8. Let $d_1 = \Pi_{i\in I}(t_i \Rightarrow p_i)$ and $d_2 = \Pi_{i\in I}(t_i \Rightarrow q_i)$ be such that $p_i, q_i \in \mathscr{S}$, $t_i \neq t_j$ for $i \neq j$ and $d_1 \sim_e d_2$. Then $t_i \Rightarrow p_i \sim_e t_i \Rightarrow q_i$ for every $i \in I$.

Proof. We proceed by induction on $\#I$. Case $\#I = 1$ follows by the hypothesis. We assume the statement for $\#I \leqslant n-1$ and prove it for $\#I = n$. Assume that the various local clocks are ordered in such a way that $t_{i_1} < \cdots < t_{i_n}$ if $I = \{i_1, \ldots, i_n\}$. Let $d' = t_{i_n} \Rightarrow p_{i_n}$, $d'' = t_{i_n} \Rightarrow q_{i_n}$, $O_{[]}' = \Pi_{i\in I-\{i_n\}}(t_i \Rightarrow p_i) \mid []$ and $O_{[]}'' = \Pi_{i\in I-\{i_n\}}(t_i \Rightarrow q_i) \mid []$. Then:

(1) $d \rhd\!\!\gg O$, with $d \in \{d', d''\}$ and $O \in \{O', O''\}$, and
(2) $O'[d'] \sim_e O''[d'']$.

Thus, by Lemma B.7, we have $t_{i_n} \Rightarrow p_{i_n} \sim_e t_{i_n} \Rightarrow q_{i_n}$ and

$$\Pi_{i\in I-\{i_n\}}(t_i \Rightarrow p_i) \sim_e \Pi_{i\in I-\{i_n\}}(t_i \Rightarrow q_i).$$

By the induction hypothesis, the main statement then follows since $\#(I - \{i_n\}) < n$. □

https://doi.org/10.1017/S0960129503004018 Published online by Cambridge University Press

Consider the structural equivalence $\equiv$ defined as the smallest congruence over states $\mathscr{D}_{crav}$ that satisfies the followings laws (where $d_1, d_2, d_3 \in \mathscr{D}_{crav}$ and $t \in \mathbb{N}$):

- $d_1 \equiv d_1 \mid (t \Rightarrow nil)$
- $t \Rightarrow (n)p \equiv t + n \Rightarrow p$
- $t \Rightarrow \text{rec } x.p \equiv t \Rightarrow p[\text{rec } x.p/x]$
- $d_1 \mid d_2 \equiv d_2 \mid d_1$
- $d_1 \mid (d_2 \mid d_3) \equiv (d_1 \mid d_2) \mid d_3$
- $(d_1 \mid d_2) \setminus B \equiv (d_1 \setminus B) \mid (d_2 \setminus B)$
- $(d_1 \mid d_2)[\Phi] \equiv (d_1[\Phi]) \mid (d_2[\Phi])$

These equalities are sound with respect to our performance-based equivalences. It has to be noted that some of them, and in particular, the last two, hold only because of the restrictions in the current language. For standard reasons, they are not sound in the presence of invisible actions.

Two important lemmas now follow. The first proves a reduction result. Each process can be reduced to the parallel composition of components that do not have delay operators at the top level.

Lemma B.9. Let $p \in \mathscr{P}_{crav}$ and $t, k \in \mathbb{N}$. Then there are $t_i \in \mathbb{N}$, $p_i \in \mathscr{S}$ such that $t_i \geqslant t$ for every i, $t_i \neq t_j$ for $i \neq j$ and

$$t + (-)k \Rightarrow p \equiv \Pi_{i \in I} \, (t_i + (-)k \Rightarrow p_i)$$

Proof. In order to use induction on the structure of p, we prove this statement for open terms. The statement for processes follows as a corollary. We proceed by case analysis on the form of p. The additional requirement $t_i \neq t_j$ for $i \neq j$ follows by the clock distribution equation $(t \Rightarrow p) \mid (t \Rightarrow q) = (t \Rightarrow p \mid q)$.

(a) $p = nil$ and $p = a.p_1$ are trivial. Set I is a singleton.

(b) $p = (n)p_1$. Then $t + (-)k \Rightarrow p = t + (-)k \Rightarrow (n)p_1 \equiv t + n + (-)k \Rightarrow p_1$. By the induction hypothesis, $t + n + (-)k \Rightarrow p_1 \equiv \Pi_{i \in I} \, (t_i + (-)k \Rightarrow p_i)$ where $t_i \geqslant t + n$. Thus, $t + (-)k \Rightarrow (n)p_1 \equiv \Pi_{i \in I} \, (t_i + (-)k \Rightarrow p_i)$ and $t_i \geqslant t + n \geqslant n$.

(c) $p = \sum_{i \in I} s_i$. This case is trivial since $\sum_{i \in I} s_i \in \mathscr{S}$. Set I is a singleton.

(d) $p = p_1 \mid p_2$. By the induction hypothesis, $t + (-)k \Rightarrow p_1 \equiv \Pi_{i \in I} \, (t_i^1 + (-)k \Rightarrow p_i^1)$, $t_i^1 \geqslant t$, and $t + (-)k \Rightarrow p_2 \equiv \Pi_{j \in J} \, (t_j^2 + (-)k \Rightarrow p_j^2)$, $t_i^2 \geqslant t$.
Then,

$$\begin{aligned} t + (-)k \Rightarrow (p_1 \mid p_2) \;\equiv\; & (\Pi_{i \in I} \, (t_i^1 + (-)k \Rightarrow p_i^1)) \mid (\Pi_{j \in J} \, (t_j^2 + (-)k \Rightarrow p_j^2)) \\ \equiv\; & \Pi_{i \in I, j \in J \text{ and } t_i^1 = t_j^2} \, (t_i^1 + (-)k \Rightarrow (p_i^1 \mid p_j^2))) \mid \\ & (\Pi_{i \in I \text{ and } t_i^1 \neq t_j^2, \forall j \in J} \, (t_i^1 + (-)k \Rightarrow p_i^1)) \mid \\ & (\Pi_{j \in J \text{ and } t_j^2 \neq t_i^1, \forall i \in I} \, (t_j^2 + (-)k \Rightarrow p_j^2)) \end{aligned}$$

(e) $p = p_1 \setminus B$. By the induction hypothesis, $t + (-)k \Rightarrow p_1 \equiv \Pi_{i \in I} \, (t_i^1 + (-)k \Rightarrow p_i^1)$, where every $t_i \geqslant t$. Then $t + (-)k \Rightarrow (p_1 \setminus B) \equiv (t + (-)k \Rightarrow p_1) \setminus B$. Finally, this latter state is $\equiv$-equivalent to $(\Pi_{i \in I} \, (t_i^1 + (-)k \Rightarrow p_i^1)) \setminus B$, which in turn is $\equiv$-equivalent to $\Pi_{i \in I} \, t_i^1 + (-)k \Rightarrow (p_i^1 \setminus B)$.

https://doi.org/10.1017/S0960129503004018 Published online by Cambridge University Press

(f) $p = p_1[\Phi]$. This case is similar to the previous one.

(g) $p = \text{rec } x.p_1$. Then $t + (-)k \Rightarrow \text{rec } x.p_1 \equiv t + (-)k \Rightarrow p_1[\text{rec } x.p_1/x]$. By the induction hypothesis,

$$\begin{aligned} t + (-)k \Rightarrow p_1[\text{rec } x.p_1/x] &= (t + (-)k \Rightarrow p_1)[\text{rec } x.p_1/x] \\ &\equiv (\Pi_{i\in I}\ (t_i + (-)k \Rightarrow p_i))[\text{rec } x.p_1/x], \end{aligned}$$

where $t_i \geqslant t$.

Now, $(\Pi_{i\in I}\ (t_i + (-)k \Rightarrow p_i))[\text{rec } x.p_1/x] \equiv (\Pi_{i\in I}\ (t_i + (-)k \Rightarrow p_i[\text{rec } x.p_1/x]))$. Finally, observe that each $p_i[\text{rec } x.p_1/x] \in \mathscr{S}$, since by the induction hypothesis, each $p_i \in \mathscr{S}$ and each p_i is guarded. □

We now prove a version of Lemma A.2 extended to timed states.

Lemma B.10. Let $t, t', r \in \mathbb{N}$, $k \in \mathbb{N}^+$, $p \in \mathscr{S}$, $d \in \mathscr{D}_{crav}$ and $a \in Act$ be such that $(t \Rightarrow p) \xrightarrow{\langle a,t',r\rangle} d$. Then, there exist $I \subseteq \mathbb{N}$, $\#I < \infty$, $t_i, t_i^1, t_i^2 \in \mathbb{N}$, $s_i \in \mathscr{S}$ for every $i \in I$, $p' \in \mathscr{S}$ such that $d \equiv (t \Rightarrow p') \mid \Pi_{i\in I}\ (t_i \Rightarrow s_i)$, $t_i \geqslant t + f(a) + r$ and:

(1) $(t \Rightarrow p) \xrightarrow{\langle a,t'-r,0\rangle} d' \equiv (t \Rightarrow p') \mid \Pi_{i\in I}\ (t_i - r \Rightarrow s_i)$ if $r > 0$

(2) $(t \Rightarrow p) \xrightarrow{\langle a,t'+k,k\rangle} d' \equiv (t \Rightarrow p') \mid \Pi_{i\in I}\ (t_i + k \Rightarrow s_i)$ if $r = 0$.

Proof. The proof is a standard induction on the depth of $(t \Rightarrow p) \xrightarrow{\langle a,t',r\rangle} d$. We show the most interesting case only; namely, action prefixing. All the other cases follow by similar reasoning.

Assume $p = a.p_1$ and $(t \Rightarrow a.p_1) \xrightarrow{\langle a,t+f(a)+r,r\rangle} t + f(a) + r \Rightarrow p_1$.

By Lemma B.9, $t + f(a) + r \Rightarrow p_1 \equiv (t + f(a) + r \Rightarrow nil) \mid \Pi_{i\in I}\ (t_i \Rightarrow s_i)$, where $s_i \in \mathscr{S}$ and $t_i \geqslant t + f(a) + r > t$. Moreover:

(1) $(t \Rightarrow a.p) \xrightarrow{\langle a,t'-r,0\rangle} t + f(a) \Rightarrow p \equiv (t \Rightarrow nil) \mid \Pi_{i\in I}\ (t_i - r \Rightarrow s_i)$ if $r > 0$

(2) $(t \Rightarrow a.p) \xrightarrow{\langle a,t'+k,k\rangle} t + f(a) + k \Rightarrow p \equiv (t \Rightarrow nil) \mid \Pi_{i\in I}\ (t_i + k \Rightarrow s_i)$ if $r = 0$. □

Another key lemma in this section says that if two timed states $t \Rightarrow p_i$ $(i \in \{1,2\})$ are eager equivalent, then $t^* \Rightarrow p_i$ $(i \in \{1,2\})$, for every $t^* > t$, are also eager equivalent. This statement requires the introduction of a new function *up*, which given a state $d \in \mathscr{D}_{crav}$ and a natural number $n \in \mathbb{N}$, returns state d where each local clock is increased of n time units.

Definition B.4. Let $up : \mathscr{D}_{crav} \times \mathbb{N} \to \mathscr{D}_{crav}$ be the least relation that satisfies the following inference rules (where $t, n \in \mathbb{N}$, $a \in Act$, $d, d_1, d_2 \in \mathscr{D}_{crav}$, and $p_i, \text{rec } x.p \in \mathscr{P}_{crav}$):

$$\begin{aligned} up(t \Rightarrow nil,\ n) &= t + n \Rightarrow nil \\ up(t \Rightarrow a.p_1,\ n) &= t + n \Rightarrow a.p_1 \\ up(t \Rightarrow (m)p_1,\ n) &= t + n \Rightarrow (m)p_1 \\ up(t \Rightarrow \textstyle\sum_{i\in I} p_i,\ n) &= t + n \Rightarrow \textstyle\sum_{i\in I} p_i \\ up(d_1 \mid d_2,\ n) &= up(d_1,\ n) \mid up(d_2,\ n) \\ up(d \setminus B,\ n) &= up(d,\ n) \setminus B \\ up(t \Rightarrow \text{rec } x.p,\ n) &= t + n \Rightarrow \text{rec } x.p \\ up(d[\Phi],\ n) &= up(d,\ n)[\Phi] \end{aligned}$$

https://doi.org/10.1017/S0960129503004018 Published online by Cambridge University Press

Eager transitions out of states $d \in \mathscr{D}_{crav}$ and the corresponding ones out of states $up(d, n)$ are strictly related. See Corradini (2000) for a proof of the following statement.

Lemma B.11. Let $d, d', d'' \in \mathscr{D}_{crav}$, $a \in Act$ and $t, n \in \mathbb{N}$. Then:

(1) $up(d, n) \xrightarrow{\langle a,t,0\rangle} d'$ implies $d \xrightarrow{\langle a,t-n,0\rangle} d''$ and $d' = up(d'', n)$

(2) $d \xrightarrow{\langle a,t,0\rangle} d'$ implies $up(d, n) \xrightarrow{\langle a,t+n,0\rangle} up(d', n)$.

Lemma B.12. Let $t \in \mathbb{N}$, $p_1, p_2 \in \mathscr{P}_{crav}$ be such that $(t \Rightarrow p_1) \sim_e (t \Rightarrow p_2)$. Then, for each $t^* > t$, we have $(t^* \Rightarrow p_1) \sim_e (t^* \Rightarrow p_2)$.

Proof. Assume $p_1, p_2 \in \mathscr{P}_{crav}$, $t, t^* \in \mathbb{N}$ be such that $(t \Rightarrow p_1) \sim_e (t \Rightarrow p_2)$ and $t^* > t$. Let $\mathscr{K}$ be the $\sim_e$-bisimulation such that $(t \Rightarrow p_1, t \Rightarrow p_2) \in \mathscr{K}$.

To prove $(t^* \Rightarrow p_1) \sim_e (t^* \Rightarrow p_2)$, we define

$$\mathfrak{R} = \{(up(d_1, n), up(d_2, n)) \mid n \in \mathbb{N} \text{ and } (d_1, d_2) \in \mathscr{K}\}$$

and prove that $\mathfrak{R}$ is an $\sim_e$-bisimulation. Clearly, it contains the pair $(t^* \Rightarrow p_1, t^* \Rightarrow p_2)$.

Consider $n \in \mathbb{N}$, $d_1, d_2 \in \mathscr{D}_{crav}$ such that $(d_1, d_2) \in \mathscr{K}$ and $(up(d_1, n), up(d_2, n)) \in \mathfrak{R}$ and assume $up(d_1, n) \xrightarrow{\langle a,t,0\rangle} d_1'$ (the symmetric case is similar). From item (1) of Lemma B.11, we have $d_1 \xrightarrow{\langle a,t-n,0\rangle} d_1''$ and $d_1' = up(d_1'', n)$. Since $(d_1, d_2) \in \mathscr{K}$, we have $d_2 \xrightarrow{\langle a,t-n,0\rangle} d_2''$ and $(d_1'', d_2'') \in \mathscr{K}$. By Lemma B.11(2), we also have $up(d_2, n) \xrightarrow{\langle a,t,0\rangle} d_2'$ and $d_2' = up(d_2'', n)$. Hence, from $(d_1'', d_2'') \in \mathscr{K}$, it is $(d_1', d_2') \in \mathfrak{R}$. □

Theorem 3.2. Let $p_1, p_2 \in \mathscr{P}_{crav}$. Then, $p_1 \sim_e p_2$ if and only if $p_1 \sim_c p_2$.

Proof. Assume $p_1, p_2 \in \mathscr{P}_{crav}$. $p_1 \sim_c p_2$ implies $p_1 \sim_e p_2$ immediately follows by Proposition 3.1. Now we assume $p_1 \sim_e p_2$ and prove $p_1 \sim_c p_2$. Let

$$\begin{aligned}\mathfrak{R} = \{(d', d'') \mid\ & d' \equiv (\Pi_{i\in I}\,(t_i^{(0)} \Rightarrow p_i^{(0)})),\ d'' \equiv (\Pi_{i\in I}\,(t_i^{(0)} \Rightarrow q_i^{(0)})) \text{ for some} \\ & I = \{1, \ldots, n\} \text{ such that } n \in \mathbb{N}^+ \text{ and, for each } i \in I, \\ & (t_i^{(0)} \Rightarrow p_i^{(0)}) \sim_e (t_i^{(0)} \Rightarrow q_i^{(0)}),\ t_i^{(0)} \in \mathbb{N},\ p_i^{(0)}, q_i^{(0)} \in \mathscr{S}\}.\end{aligned}$$

We then prove that $\mathfrak{R}$ is a PC-bisimulation. By Lemma B.9, $0 \Rightarrow p_1 \equiv \Pi_{i\in I}\,(t_i \Rightarrow p_i^1)$ and $0 \Rightarrow p_2 \equiv \Pi_{j\in J}\,(t_j \Rightarrow p_j^2)$. By $d \equiv d \,|\, (t \Rightarrow nil)$, and the associativity and commutativity of parallel composition, we can add parallel components (of the form $t \Rightarrow nil$) in such a way that $I = J$ and every local clock that appears in $\Pi_{i\in I}\,(t_i \Rightarrow p_i^1)$ also appears in $\Pi_{j\in J}\,(t_j \Rightarrow p_j^2)$, and *vice versa*. Then we can group parallel components with the same local clock, by applying $(t \Rightarrow p) \,|\, (t \Rightarrow q) = t \Rightarrow (p \,|\, q)$ from right to left. Then, Lemma B.8 can be used to state that $(0 \Rightarrow p_1, 0 \Rightarrow p_2) \in \mathfrak{R}$.

Take a generic $(d', d'') \in \mathfrak{R}$. Then $d' \equiv \Pi_{i\in I}\,(t_i^{(0)} \Rightarrow p_i^{(0)})$ and $d'' \equiv \Pi_{i\in I}\,(t_i^{(0)} \Rightarrow q_i^{(0)})$, where $I = \{1, \ldots, n\}$ and $t_i^{(0)} \in \mathbb{N}$, $p_i^{(0)}, q_i^{(0)} \in \mathscr{S}$, $(t_i^{(0)} \Rightarrow p_i^{(0)}) \sim_e (t_i^{(0)} \Rightarrow q_i^{(0)})$, for each $i \in I$.

Assume $d' \equiv \Pi_{i\in I}\,(t_i^{(0)} \Rightarrow p_i^{(0)}) \xrightarrow{\langle a,t,r\rangle} d_1$ where $r \geqslant 0$ (the symmetric case is similar). Then there exists $k \in I$ such that $(t_k^{(0)} \Rightarrow p_k^{(0)}) \xrightarrow{\langle a,t,r\rangle} d_2$ and $d_1 = \Pi_{i\in I}\,(d_i')$, where $d_k' = d_2$, $d_i' = (t_i^{(0)} \Rightarrow p_i^{(0)})$ for each $i \neq k$. Assume $r > 0$ (the case $r = 0$ is similar). Lemma B.10 applied to transition $(t_k^{(0)} \Rightarrow p_k^{(0)}) \xrightarrow{\langle a,t,r\rangle} d_2$ gives $d_2 \equiv (t_k^{(0)} \Rightarrow p') \,|\, \Pi_{m\in M}\,(t_m^1 \Rightarrow p_m^1)$, $t_m^1 \geqslant$

https://doi.org/10.1017/S0960129503004018 Published online by Cambridge University Press

$t_k^{(0)} + f(a) + r$. Again by Lemma B.10, we also have $(t_k^{(0)} \Rightarrow p_k^{(0)}) \xrightarrow{\langle a, t-r, 0\rangle} d_3$ and $d_3 \equiv (t_k^{(0)} \Rightarrow p') \mid \Pi_{m\in M}\,(t_m^1 - r \Rightarrow p_m^1)$.

Using the latter transition and the hypothesis $(t_k^{(0)} \Rightarrow p_k^{(0)}) \sim_e (t_k^{(0)} \Rightarrow q_k^{(0)})$, we have $(t_k^{(0)} \Rightarrow q_k^{(0)}) \xrightarrow{\langle a, t-r, 0\rangle} d_4$ and $d_3 \sim_e d_4$. By Lemma B.10, $d_4 \equiv (t_k^{(0)} \Rightarrow q') \mid \Pi_{o\in O}\,(t_o^2 \Rightarrow q_o^2)$, where $t_o^2 \geqslant t_k^{(0)} + f(a)$. Moreover, again by Lemma B.10, $(t_k^{(0)} \Rightarrow q_k^{(0)}) \xrightarrow{\langle a, t, r\rangle} d_5$ and $d_5 \equiv (t_k^{(0)} \Rightarrow q') \mid \Pi_{o\in O}\,(t_o^2 + r \Rightarrow q_o^2)$ (thus $t_o^2 + r \geqslant t_k^{(0)} + f(a) + r$).

Thus, we also have $d'' \equiv \Pi_{i\in I}\,(t_i^{(0)} \Rightarrow q_i^{(0)}) \xrightarrow{\langle a, t, r\rangle} d_6 = \Pi_{i\in I}\,(d_i'')$ where $d_k'' = d_5$ and $d_i'' = (t_i^{(0)} \Rightarrow q_i^{(0)})$ for each $i \neq k$.

The rest of the proof is devoted to proving that $(d_1, d_6) \in \mathfrak{R}$. To do this, it is sufficient to prove that d_2 and d_5 can be decomposed into $\sim_e$-equivalent components as required by relation $\mathfrak{R}$.

Regarding d_3 and d_4, by rule $d \equiv d \mid (t \Rightarrow nil)$, and the associativity and commutativity of parallel composition, we can assume, as above, that $M = O$ and every local clock that appears in $\Pi_{m\in M}\,(t_m^1 - r \Rightarrow p_m^1)$ also appears in $\Pi_{o\in O}\,(t_o^2 \Rightarrow q_o^2)$, and *vice versa*. Of course, we can apply the same rules to make sure that the same properties also apply in $\Pi_{m\in M}\,(t_m^1 \Rightarrow p_m^1)$ and $\Pi_{o\in O}\,(t_o^2 + r \Rightarrow q_o^2)$ appearing within the $\equiv$-equivalent forms of d_2 and d_5, respectively. Actually we can additionally assume that $t \Rightarrow p_m^1$ and $t \Rightarrow q_o^2$ appear within d_3 and d_4 (within their $\equiv$-equivalent forms) if and only if $t + r \Rightarrow p_m^1$ and $t + r \Rightarrow q_o^2$ appear within d_2 and d_5 (simply apply $d \equiv d \mid (t + r \Rightarrow nil)$ in d_2 and d_5 whenever $d \equiv d \mid (t \Rightarrow nil)$ is applied in d_3 and d_4). Hence, without loss of generality, we can assume:

$$
\begin{aligned}
d_3 &\equiv (t_k^{(0)} \Rightarrow p') \mid \Pi_{m\in M}\,(t_m \Rightarrow p_m^1),\\
d_4 &\equiv (t_k^{(0)} \Rightarrow q') \mid \Pi_{m\in M}\,(t_m \Rightarrow q_m^2)\\
d_2 &\equiv (t_k^{(0)} \Rightarrow p') \mid \Pi_{m\in M}\,(t_m + r \Rightarrow p_m^1)\\
d_5 &\equiv (t_k^{(0)} \Rightarrow q') \mid \Pi_{m\in M}\,(t_m + r \Rightarrow q_m^2).
\end{aligned}
$$

By hypothesis, $d_3 \sim_e d_4$ and $t_m > t_k^{(0)}$. Since p', q', p_m^1 and q_m^2 are in $\mathscr{S}$, by Lemma B.8, we have $t_k^{(0)} \Rightarrow p' \sim_e t_k^{(0)} \Rightarrow q'$ and $t_m \Rightarrow p_m^1 \sim_e t_m \Rightarrow q_m^2$. By Lemma B.12, we also have $t_m + r \Rightarrow p_m^1 \sim_e t_m + r \Rightarrow q_m^2$. Hence $(d_2, d_5) \in \mathfrak{R}$, and, consequently, $(d_1, d_6) \in \mathfrak{R}$ also. □

Acknowledgments

We would thank the anonymous referees for their helpful comments.

References

Aceto, L. and Murphy, D. (1996) Timing and Causality in Process Algebra. *Acta Informatica* **33** (4) 317–350.

Baeten, J. and Bergstra, J. (1991) Real time process algebra. *Formal Aspects of Computing* **3** (2) 142–188.

Bérard, B., Labroue, A. and Schnoebelen, Ph. (2000) Verifying performance equivalence for Timed Basic Parallel Processes. In: FOSSACS 2000. *Springer-Verlag Lecture Notes in Computer Science* **1784** 35–47.

https://doi.org/10.1017/S0960129503004018 Published online by Cambridge University Press

Corradini, F. (1998) On Performance Congruences for Process Algebras. *Information and Computation* **145** 191–230.

Corradini, F. (2000) Absolute versus Relative Time in Process Algebras. *Information and Computation* **156** (1) 122–172. (An extended abstract of this paper with the same title appeared in EXPRESS'97. *Electronic Notes of Theoretical Computer Science*, 1997.)

Corradini, F., Ferrari, G. L. and Pistore, M. (2001) On the Semantics of Durational Actions. *Theoretical Computer Science* **269** 47–82. (An extended abstract of this paper with the title 'Eager, Busy-Waiting and Lazy Actions in Timed Computation' appeared in EXPRESS'97. *Electronic Notes of Theoretical Computer Science*, 1997.)

Corradini, F. and Di Cola, D. (2000) The Expressive Power of Urgent, Lazy and Busy-Waiting Actions in Timed Processes (Extended Abstract). In: EXPRESS'00. *Electronic Notes of Theoretical Computer Science*, 2000.

Corradini, F. and Di Cola, D. (2001) On Testing Urgency through Laziness over Processes with Durational Actions. *Theoretical Computer Science* **258** 393–407.

Ferrari, G.-L. and Montanari, U. (1995) Dynamic matrices and the cost analysis of concurrent programs. In: AMAST'95. *Springer-Verlag Lecture Notes in Computer Science* **936** 307–321.

Gorrieri, R., Roccetti, M. and Stancampiano, E. (1995) A Theory of Processes with Durational Actions. *Theoretical Computer Science* **140** (1) 73–94.

Hennessy, M. and Regan, T. (1995) A Process Algebra for Timed Systems. *Information and Computation* **117** 221–239.

Milner, R. (1989) *Communication and Concurrency*, International series on computer science, Prentice Hall International.

Moller, F. and Tofts, C. (1990) A Temporal Calculus of Communicating Systems. In: CONCUR'90. *Springer-Verlag Lecture Notes in Computer Science* **459** 401–415.

Nicollin, X. and Sifakis, J. (1991) On overview and synthesis on timed process algebras. In: Real Time: Theory in Practice. *Springer-Verlag Lecture Notes in Computer Science* **600** 526–548.

Nicollin, X. and Sifakis, J. (1994) The algebra of timed processes ATP: theory and applications. *Information and Computation* **114** 131–178.

Reed, G. M. and Roscoe, A. W. (1988) A timed model for communicating sequential processes. *Theoretical Computer Science* **58** 249–261.

Yi, W. (1990) Real time behaviour of asynchronous agents. In: CONCUR'90. *Springer-Verlag Lecture Notes in Computer Science* **458** 502–520.

https://doi.org/10.1017/S0960129503004018 Published online by Cambridge University Press