A Study on the Specification and Verification of Performance Properties* (Extended Abstract)

Xiao Jun Chen[1], Flavio Corradini[2], Roberto Gorrieri[3]

[1] Dipartimento di Scienze dell'Informazione, Università "La Sapienza"
e-mail: chen@dsi.uniroma1.it
[2] School of Cognitive and Computing Sciences, University of Sussex
e-mail: flavioc@cogs.sussex.ac.uk
[3] Dipartimento di Scienze dell'Informazione, Università di Bologna
e-mail: gorrieri@cs.unibo.it

Abstract. The process algebra expressing *nets of automata* is provided with a new operational semantics, that allows us to reason about performance properties of concurrent systems. Based on the idea that actions are durational and execution is asynchronous, performance equivalence and performance preorder are introduced as extensions of *interleaving bisimulation.* They turn out to be a congruence and a precongruence, respectively, w.r.t. the operators of the language we study. The equivalence (preorder) is also characterized by Hennessy-Milner logic (and its positive form), with natural numbers as propositional constants. Furthermore, we show how to apply modal μ-calculus to specify and verify performance-oriented properties.

1 Introduction

The analysis of system efficiency can be done in process algebras by way of suitable notions of equivalences and preorders [1, 3, 13, 18]. Some of them [1, 2, 8, 9, 11, 12, 13] use the time consumed for process execution as a measure of the efficiency: they rely on the assumption that the actions have a *static duration*, chosen on the basis of the features of the abstract machine. Precisely, systems are distributed over the space where each sequential component is associated with a *local clock* whose elapsing is set dynamically during the execution of the actions by the corresponding component. Whenever an action a is executed by a sequential component E, the value n of the clock of E is incremented to n plus the duration of a, whilst the local clocks of those sequential components not involved in the execution of a are unaffected. By incorporating the duration of actions into the notion of process equivalence/preorder, we are able to discriminate processes not only according to the *functionality* (what actions a process can do), but also according to their *performance*.

* This work has been partially founded by EEC within the HCM Project EXPRESS, and by CNR.

In [2, 13], *performance equivalences* are introduced, based on the standard notion of *bisimulation* [17]: processes are performance equivalent whenever they perform the same actions in the same amount of time. The approach of [2], however, is rather strict because it requires a very stringent synchronization rule, preventing possible synchronizations that need some busy waiting from one of the communicating partners. Moreover, some synchronizations that are possible in the interleaving "untimed" case are actually prevented; thus, performance equivalence on the resulting transition system is not a refinement of interleaving behavioural equivalences. Another approach, relaxing the synchronization rule to allow any form of busy waiting in a synchronization, is taken in [13]; the resulting semantics is comparable with the interleaving untimed semantics, but has the unfortunate effect of loosing compositionality. Both proposals assume urgency on actions (unless in a synchronization context for the latter proposal) and both admit the phenomenon of the so-called "well-caused but ill-timed" executions [2].

In this paper, we present a new operational semantics for a process algebra expressing nets of automata. Differently from the above, we allow actions to be executed in delay in any context (as in [7]), resulting in a rather simple operational semantics for the synchronization among processes. However, actions cannot be delayed at will, but there is an upper bound due to the maximum local clock value, representing the value of the global clock. This essentially guarantees that the resulting labelled transition system is *finitely branching*. Thus, our point of view lies between *eager* systems [1, 2, 8, 9, 11, 12, 13], where actions have to be performed as soon as possible,[4] and *lazy* systems [7] where systems can delay the execution of actions arbitrarily long. Our approach has also the effect of forbidding the generation of ill-timed traces, where the "observation time" (the order of transitions generation) agrees with the "execution time" (the number occurring in the label of a transition); in other words, execution time in a computation always increases. Finally, no interleaving (untimed) synchronization is prevented, so that performance equivalence is in our case more discriminating than interleaving bisimulation equivalence. Based on this new transitional semantics, (i) performance equivalence is a congruence; (ii) moreover, performance preorder, which *orders processes with respect to speed*, is a precongruence. Intuitively, a process E_1 is faster than another process E_2 iff they are bisimilar (i.e. they have the same functional behaviour), and E_1 is at least as fast as E_2. This preorder extends the notion in [8, 9] to deal with communicating processes under the hypothesis of lazyness of actions.

Equivalences and preorders can also be characterized by means of modal and temporal logics: two processes are equivalent iff they are not distinguishable by formulas of the logic, and a process q is finer than p iff q satisfies all formulas that p does. As it is well-known, interleaving bisimulation equivalence is fully characterized by Henessy-Milner logic [14] (HML for short). In this paper, we show that performance equivalence (as defined above) can be logically charac-

[4] In [12, 13] actions may be delayed at will only if they are involved in a synchronization with a late partner.

terized by suitably extending HML with an additional set of propositions, i.e. the natural numbers. Intuitively, these numbers are used to test the passing of the time during the possible computations of running systems. We give interpretation of these propositions on labelled transition systems, and show that with this interpretation, the equivalence deduced by this logic coincides with performance equivalence. Furthermore, the positive form of HML is also extended with natural numbers (together with a new suitable interpretation for them), based on which the performance preorder mentioned above can be fully characterized.

In comparison with process algebras, logics have the advantage of not always requiring to specify the full behavior of a system: they can be used to focus on specific interesting properties of systems, like safety or fairness. As the expressivity of HML is rather limited, we adopt modal μ-calculus [15] to specify performance properties; essentially, we consider HML extended by fixed point operators. As above, the natural numbers are taken as propositional constants.

If the process is finite state, properties expressed in modal μ-calculus can be verified in polynomial time [5] and the verification can be done automatically by means of tools like the Concurrency Workbench [6]. In our approach the number of states of labelled transition system for a process is infinite in general, because states contain clocks, the value of which increases as the computation proceeds. We present a technique for reducing the check of a property ϕ on an infinite state labelled transition system TS to the check of a different property $F(\phi)$ on a finite state approximation TS' of TS, thus allowing to exploit existing verification tools on finite state processes.

The paper is organized as follows: section 2 presents the syntax, the operational semantics of the language and the notions of performance equivalence/preorder; section 3 shows the logical characterizations of the performance equivalence/preorder, and section 4 discusses the way to specify and verify performance properties using modal μ-calculus.

2 The Language and its Behavioural Semantics

In this section, we present the algebraic description of the processes with durational actions, including the syntax, the operational semantics, and the notions of performance equivalence/preorder.

2.1 The Language

We consider *timed processes* as *nets of automata* equipped with clocks. Nets are obtained by composing regular processes (sequential and non-deterministic processes) by means of the static operators of CCS [17]. Thus, the number of states the classic interleaving semantics associates to each net is finite.

Let A (ranged over by a, b, $\ldots$) be a set of atomic actions, $\bar{A}$ (ranged over by $\bar{a}$, $\bar{b}$, ...) its complementation, and Act (ranged over by α, β, ...) defined as $Act = A \cup \overline{A}$. Complementation is extended to Act by $\overline{\overline{\alpha}} = \alpha$. As usual, τ

($\tau \notin Act$) denotes the invisible or internal action, and we use Act_τ to denote $Act \cup \{\tau\}$.

Let V (ranged over by $v, v', v'', \ldots.$) denote a set of variables. The set of regular processes $\mathcal{P}$ ($p,\ q \in \mathcal{P}$) over Act and V, is the set of closed (i.e., without free variables) and guarded (i.e., variables appear within a $a.\_$ or wait $n.\_$ context) terms generated by the following grammar:

$$p\ ::=\ nil \mid \alpha.p \mid \text{wait } n.\ p \mid p + p \mid v \mid \text{rec } v.\ p$$

Process nil denotes a terminated process. By prefixing a term p with an action α, we get a process term $\alpha.p$ which can do an action α and then behaves like p. wait $n.\ p$ denotes a process which waits for at least $n \in N^+$ time units and then behaves like process p. $p_1 + p_2$ denotes the alternative composition of p_1 and p_2, while rec $v.\ p$ is used for recursive definitions.

The set $\mathcal{E}$ of *nets of automata* ($E,\ E' \in \mathcal{E}$) is defined by the following grammar:

$$E\ ::=\ p \mid E|E \mid E[\Phi] \mid E\backslash\{\alpha\}$$

$E_1|E_2$, the parallel composition of E_1 and E_2, is the process which can perform any interleaving of the actions of E_1 and E_2 or synchronizations whenever E_1 and E_2 can perform complementary actions. $E[\Phi]$ is a process which behaves like E but its actions are relabelled according to function Φ. Finally, $E\backslash\{\alpha\}$ is a process which behaves like E but actions $\alpha, \bar{\alpha}$ are forbidden.

The states of the labelled transition system we are going to define are processes enriched with local clocks which record the local view of the elapsing of time together with the value of the global clock (corresponding to the maximum of local views). Formally, we introduce extended processes, obtained by *nets of automata* with a *local clock prefixing* operator, $n \Rightarrow \_$:

$$d\ ::=\ n \Rightarrow p \mid d|d' \mid d\backslash\{\alpha\} \mid d[\Phi].$$

with $p \in \mathcal{P}$, $n \in N$. We use $\mathcal{D}$ (ranged over by $d,\ d'\ \ldots$) to denote the set of terms generated by this grammar. The *clock distribution equations*

$$n \Rightarrow (E \mid E') = (n \Rightarrow E) \mid (n \Rightarrow E')$$

$$n \Rightarrow (E\backslash\{\alpha\}) = (n \Rightarrow E)\backslash\{\alpha\}$$

$$n \Rightarrow (E[\Phi]) = (n \Rightarrow E)[\Phi]$$

show that a term $n \Rightarrow E$ can be reduced canonically to an extended process, when interpreting these equations as rewrite rules from left to right.

A *timed process* (or state) $d \triangleright n$ is an extended process d, equipped with a global clock n. The global clock represents the global observation time for the execution. The explicit use of a global clock allows us to define a semantics that avoids the system to stay idle. This will be clearer in the next subsection. Here, we require that $n \geq maxclock(d)$, where $maxclock(d)$ gives the maximum of clock values m occurring in subterms of d of the form $m \Rightarrow p$. We use $\mathcal{S}$ (ranged over by $s,\ t, \ldots$) to denote the set of timed processes.

The amount of time needed for the execution of a particular action is fixed once for all on the basis of the chosen machine. As the duration of actions can be different for different machines, we should take this parameter into account in our semantic description. One possibility is to introduce *duration functions* (ranged over by f, g) which associate to each action a the positive integer $f(a)$ of time units needed for its execution. In the following, we choose a duration function f to fix this parameter.

2.2 The Operational Semantics

The *timed processes* are interpreted on *timed labelled transition systems* $\langle S, M, T \rangle$, where S is the set of *timed processes*, M a set of *timed actions*. Here, the *timed action* has form (μ, n) where $\mu \in Act_\tau$ and $n \in N^+$. Intuitively, a label (μ, n) means that action $\mu \in Act_\tau$ has been completed exactly n time units after the computation began. $T = \{\xrightarrow{\mu} \subseteq S \times S \mid \mu \in M\}$ is the *transition relation*. We will write $s \xrightarrow{\mu} s'$ instead of $\langle s, s' \rangle \in \xrightarrow{\mu}$.

As usual, the transition relation is given through a set of inference rules, listed in Table 1. It is worthwhile observing that these rules are parametric w.r.t. the chosen duraction function f. Hence, we should write $\rightarrow_f$. For the sake of simplicity, the subscript will be omitted as clear from the context.

A few comments on the rules in Table 1 are now in order. The rule for action prefixing *Act* states that action α can only be completed at a time greater than or equal to the global clock, but less than or equal to the global clock plus the duration of α. This guarantees that the system *as a whole* never stays idle. Moreover, an obvious constraint on the local clock is imposed: the time of completion is greater than (or equal to) the sum of the local clock value with the duration of α. For the last rebelling rule, we confine ourselves in this paper to duration-preserving relabellings, i.e. $f(\alpha) = f(\Phi(\alpha))$. The other rules are self explanatory.

Some properties of the operational semantics immediately follow. First we show that the time increases as the computation proceeds, so that only "well timed traces" are taken into account. Moreover, we show that the system as a whole never idles. Then we give a few examples in order to show how process computations proceed. Finally, we prove that our semantics never gets an infinitely branching transition system.

In a derivation $d \triangleright n \xrightarrow{(\mu,t)} d' \triangleright n'$, $duration(\mu) = f(a)$ if μ is a visible action a; if $\mu = \tau$, then $duration(\mu) = f(a)$ if an $(a, \bar{a})$-communication is performed; $duration(\mu) = m$ if a *wait* m is performed. Clearly, in these two latter cases, an inspection of the proof of the derivation is needed in order to solve ambiguities.

Proposition 1. *Let $d \triangleright n \xrightarrow{(\mu,t)} d' \triangleright n'$ be derivable. Then, the following hold:*

*(i) $t = n'$ and $n \leq t$ (*no ill-timed*); and*
*(ii) $n' \leq n + duration(\mu)$ (*no idling, as a whole*).*

Example 1. Let $f(\alpha) = 2,\ f(\beta) = 3$ for $\alpha,\ \beta \in Act$ and consider process $\alpha|\beta$.

- (i) The fastest terminating computation performed by $(0 \Rightarrow \alpha|0 \Rightarrow \beta) \triangleright 0$ is the one in which action α is observed at time $f(\alpha)$ and then action β is observed at time $f(\beta)$. This is possible when the two processors in which the two components are located start their execution at time 0. This corresponds to having two processors available from the beginning.
- (ii) The slowest terminating computations are the two in which the two actions are sequentialized in time. These computations take $f(\alpha) + f(\beta)$ time units to be performed. This corresponds to having only one processor available.
- (iii) Between these two ($f(\beta)$ and $f(\alpha)+f(\beta)$) extreme performance limits, there are some other possible computations. An interesting one is that when action β starts at time 0 and then a new processor is available from time 2, permitting to overlap for one time unit the execution of the two actions. Then the whole process ends at time 4.

Proposition 2. *Let $d \triangleright n$ be any timed process. Then, set*

$\{d' \triangleright n' | d \triangleright n \xrightarrow{(\mu,n')} d' \triangleright n'\}$ *is finite.*

The following example shows that we are not taking the so-called "maximal progress" approach, according to which an immediately executable action always takes precedence over actions with a later enabling time.

Example 2. Consider net $(\alpha.\beta.\delta|(\bar{\beta}+\gamma))\backslash\{\beta\}$. According to the maximal progress approach, if $f(\alpha) \geq f(\gamma)$, then action δ will never be performed. In our approach, instead, because actions are lazy, γ is not forced to be performed starting at time 0. Thus we allow an α-labelled transition at completing time $f(\alpha)$, then a communication is possible that leads to the execution of action δ at completing time $f(\alpha) + f(\beta) + f(\delta)$.

2.3 Behavioural Semantics

The observational semantics we are interested in is based on the branching-time semantics of bisimulation. We first define our notion of *performance equivalence* that relates two systems whenever they perform the same actions at the same time. Then we define *performance preoder* that allows us to order processes with respect to speed.

Definition 3. *(Performance Equivalence)*

Let Rel denote the set of binary relations over $\mathcal{S}$. The functional $R : Rel \longrightarrow Rel$ is defined, for each $\Re \in Rel$, as follows: $(d_1 \triangleright n, d_2 \triangleright n) \in R(\Re)$ if, for each $\mu \in Act_\tau$,

i) $d_1 \triangleright n \xrightarrow{(\mu,n')} d_1' \triangleright n'$ implies $d_2 \triangleright n \xrightarrow{(\mu,n')} d_2' \triangleright n'$ such that $(d_1' \triangleright n', d_2' \triangleright n') \in \Re$;

ii) $d_2 \rhd n \xrightarrow{(\mu,n')} d_2' \rhd n'$ implies $d_1 \rhd n \xrightarrow{(\mu,n')} d_1' \rhd n'$ such that $(d_1' \rhd n', d_2' \rhd n') \in \Re$.

A relation $\Re \in Rel$ will be called a *R-bisimulation* if $\Re \subseteq R(\Re)$. We say that two timed processes s_1 and s_2 are *f-performance equivalent*, notation $s_1 \sim^f s_2$, iff there exists a *R-bisimulation* $\Re$ such that $(s_1, s_2) \in \Re$. We say that two processes $E_1,\ E_2 \in \mathcal{E}$ are *f-performance equivalent*, notation $E_1 \sim_p^f E_2$, iff $(0 \Rightarrow E_1) \rhd 0 \sim^f (0 \Rightarrow E_2) \rhd 0$. When f is clear from the context, we omit the superscript f, as in $d_1 \rhd n \sim d_2 \rhd n$ and $E_1 \sim_p E_2$.

Proposition 4. *Let* $p_1,\ p_2,\ p \in \mathcal{P},\ E_1,\ E_2,\ E \in \mathcal{E},\ n \in N^+$ *and* $p_1 \sim_p p_2$. *If* $E_1 \sim_p E_2$ *then the following holds:*

$$\begin{array}{lll} \alpha.p_1 \sim_p \alpha.p_2 & wait\ n.p_1 \sim_p wait\ n.p_2 & p_1\ +\ p \sim_p p_2\ +\ p \\ E_1 \mid E \sim_p E_2 \mid E & E_1[\Phi] \sim_p E_2[\Phi] & E_1 \backslash \{\alpha\} \sim_p E_2 \backslash \{\alpha\} \end{array}$$

Moreover, like in [17], it can be proved that $\sim_p$ is a congruence for recursion as well, on automata.

While performance equivalence asks whether two processes can do the same action at the same time, performance preorder allows us to express whether two processes have the same functional behaviors but one is faster than another according to duration function f.

Definition 5. *(Performance Preorder)*

Let Rel denote the set of binary relations over $\mathcal{S}$. The functional $G : Rel \rightarrow Rel$ is defined, for each $\Re \in Rel$, as follows: $(d_1 \rhd n', d_2 \rhd n'') \in G(\Re)$ if, for each $\mu \in Act_\tau$,

i) $d_1 \rhd n' \xrightarrow{(\mu,n)} d_1' \rhd n$ implies $d_2 \rhd n'' \xrightarrow{(\mu,m)} d_2' \rhd m$ with $n \leq m$ and $(d_1' \rhd n,\ d_2' \rhd m) \in \Re$;

ii) $d_2 \rhd n'' \xrightarrow{(\mu,m)} d_2' \rhd m$ implies $d_1 \rhd n' \xrightarrow{(\mu,n)} d_1' \rhd n$ with $n \leq m$ and $(d_1' \rhd n,\ d_2' \rhd m) \in \Re$;

A relation $\Re \in Rel$ will be called a *G-bisimulation* if $\Re \subseteq G(\Re)$. We say that two timed processes s_1 and s_2 are in the relation of *f-performance preorder* (denoted by $s_1 \sqsubseteq^f s_2$) iff there exists a *G-bisimulation* $\Re$ such that $(s_1, s_2) \in \Re$. We say that $E_1 \sqsubseteq_p^f E_2$ iff $0 \Rightarrow E_1 \rhd 0 \sqsubseteq^f 0 \Rightarrow E_2 \rhd 0$. Whenever clear from the context we will omit the superscript duration function.

Example 3. $\alpha \mid \beta \sqsubseteq_p \alpha.\beta + \beta.\alpha$ but not $\alpha.\beta + \beta.\alpha \sqsubseteq_p \alpha \mid \beta$. In other words, $\alpha \mid \beta$ is faster than its interleaving explanation $\alpha.\beta + \beta.\alpha$. To show $\alpha \mid \beta \sqsubseteq_p \alpha.\beta + \beta.\alpha$, it is sufficient to observe that relation:

$$\begin{aligned}\Re = \{ & \langle (0 \Rightarrow \alpha | 0 \Rightarrow \beta) \triangleright 0, (0 \Rightarrow \alpha.\beta + \beta.\alpha) \triangleright 0 \rangle, \\ & \langle (0 \Rightarrow \alpha.\beta + \beta.\alpha) \triangleright 0, (0 \Rightarrow \alpha | 0 \Rightarrow \beta) \triangleright 0 \rangle, \\ & \langle (n \Rightarrow nil | 0 \Rightarrow \beta) \triangleright n, (n \Rightarrow \beta) \triangleright n \rangle, \\ & \langle (n \Rightarrow \beta) \triangleright n, (n \Rightarrow nil | 0 \Rightarrow \beta) \triangleright n \rangle, \\ & \langle (0 \Rightarrow \alpha | m \Rightarrow nil) \triangleright m, (m \Rightarrow \alpha) \triangleright m \rangle, \\ & \langle (m \Rightarrow \alpha) \triangleright m, (0 \Rightarrow \alpha | m \Rightarrow nil) \triangleright m \rangle, \\ & \langle (n \Rightarrow nil | m \Rightarrow nil) \triangleright max\{n, m\}, (m + n \Rightarrow nil) \triangleright n + m \rangle, \\ & \langle (m + n \Rightarrow nil) \triangleright n + m, (n \Rightarrow nil | m \Rightarrow nil) \triangleright max\{n, m\} \rangle \}\end{aligned}$$

is a *G-bisimulation*. To prove that $\alpha.\beta + \beta.\alpha \not\sqsubseteq_p \alpha \mid \beta$, it is sufficient to observe that the two transitions

$$(0 \Rightarrow \alpha \mid 0 \Rightarrow \beta) \triangleright 0 \xrightarrow{(\alpha, f(\alpha))} (f(\alpha) \Rightarrow nil \mid 0 \Rightarrow \beta) \triangleright f(\alpha)$$

$$(f(\alpha) \Rightarrow nil \mid 0 \Rightarrow \beta) \triangleright f(\alpha) \xrightarrow{(\beta, t)} (f(\alpha) \Rightarrow nil \mid t \Rightarrow nil) \triangleright t$$

with $t = max\{f(\alpha), f(\beta)\}$ cannot be matched by system $(0 \Rightarrow \alpha.\beta + \beta.\alpha) \triangleright 0$.

Now we establish a precongruence result for our performance preorder.

Proposition 6. *Let $p_1,\ p_2,\ p \in \mathcal{P},\ n \in N^+$ and $p_1 \sqsubseteq_p p_2$. Let $E_1,\ E_2,\ E \in \mathcal{E}$, Φ be a duration preserving relabelling function, and assume $E_1 \sqsubseteq_p E_2$. Then, the following holds:*

$\alpha.p_1 \sqsubseteq_p \alpha.p_2$	$wait\ n.p_1 \sqsubseteq_p wait\ n.p_2$	$p_1 + p \sqsubseteq_p p_2 + p$
$E_1 \mid E \sqsubseteq_p E_2 \mid E$	$E_1[\Phi] \sqsubseteq_p E_2[\Phi]$	$E_1 \backslash \{\alpha\} \sqsubseteq_p E_2 \backslash \{\alpha\}$

We finish this section by observing that performance semantics is consistent with classic untimed interleaving strong bisimulation. Let $\sim$ denote strong bisimulation equivalence [17] over the transition system of net of automata (label τ is used for prefixes of the form wait n).

Proposition 7. *Let $E_1,\ E_2,\ E \in \mathcal{E}$. Then, the following hold:*

(i) $E_1 \sim_p E_2$ *implies* $E_1 \sim E_2$;
(ii) $E_1 \sqsubseteq_p E_2$ *implies* $E_1 \sim E_2$.

3 Logical Characterizations

Temporal and modal logics offer another way to characterize equivalences and preorders [4, 14]: (i) two processes are equivalent if they are not distinguishable by formulas of the logical language under consideration. For example, HML gives a logical characterization of bisimulation [14], while the equivalence deduced by HML+Until coincides with branching bisimulation [19]. (ii) process Q is finer/better than P if Q satisfies all formulas that P satisfies.

In this section, we extend HML (and its positive form) with natural numbers as propositional constants. We give proper interpretations of these propositional constants on timed labelled transition systems, so that together with these constants, the equivalence deduced by HML (and its positive form), coincides with performance equivalence (performance preorder).

3.1 Characterizing Performance Equivalence

We consider the following HML formulas with natural numbers as propositional letters ($\alpha \in Act$, $n \in N^+$). For convenience, we use L_t to denote this set of formulae.

$$\phi \ ::= \ tt \mid \neg\phi \mid \phi \wedge \phi \mid \langle\alpha\rangle\phi \mid n$$

Here $\langle\alpha\rangle$ is the *next* modality as in HML, except that it is interpreted on timed labelled transition systems:

$$s \models \langle\alpha\rangle\phi \textit{ iff } \exists n, s'. \text{ such that } s \xrightarrow{(\alpha,n)} s' \text{ and } s' \models \phi$$

Remark. Formally, the satisfaction relation $\models$ is defined between a pair (TS, s) (composed of a transition system TS and a state s) and a logical formula φ. In the following, when there is no confusion, we will simply use $s \models \varphi$ instead of $(TS, s) \models \varphi$.

The propositional constants are used to identify the global clock in a state:

$$d \triangleright n' \models n \textit{ iff } n' = n$$

The equivalence deduced by L_t coincides with performance equivalence:

Proposition 8. *Let* $s_1,\ s_2 \in \mathcal{S}$. $s_1 \sim_p s_2$ *iff* $(\forall\phi \in L_t,\ s_1 \models \phi \text{ iff } s_2 \models \phi)$.

Immediately from this proposition, we have

Theorem 9. *Let* $E_1,\ E_2 \in \mathcal{P}$. $E_1 \sim_p E_2$ *iff*

$$(\forall\phi \in L_t,\ 0 \Rightarrow E_1 \triangleright 0 \models \phi \text{ iff } 0 \Rightarrow E_2 \triangleright 0 \models \phi).$$

3.2 Characterizing Performance Preorder

To characterize preorders, the positive form of the logic languages are usually adopted. In the following, we will use L_t^p to denote the set of positive HML formulas, with propositional constants. This logic is generated by the following grammar ($\alpha \in Act$, $n \in N$):

$$\phi \ ::= \ tt \mid ff \mid \phi \wedge \phi \mid \phi \vee \phi \mid \langle\alpha\rangle\phi \mid [\alpha]\phi \mid n$$

Here ff $(\neg tt)$ is the dual of tt, and $[\alpha]\phi$ $(\neg\langle\alpha\rangle\neg\phi)$ is the dual of $\langle\alpha\rangle\phi$. So, the satisfaction relation of $[\alpha]\phi$ on timed labelled transition systems is

$$s \models [\alpha]\phi \textit{ iff } \forall n, s'.\ s \xrightarrow{(\alpha,n)} s' \text{ implies } s' \models \phi$$

However, with the interpretation of the propositional constants previously defined for L_t, we have that L_t^p does not characterize performance preorder, as the following example shows.

Example 4. From Example 3, we know that $\alpha \mid \beta \sqsubseteq_p \alpha.\beta + \beta.\alpha$. But using the previous interpretation of L_t, let $\varphi = \langle\alpha\rangle\langle\beta\rangle max\{f(\alpha), f(\beta)\}$, we have $\alpha \mid \beta \models \varphi$ but $\alpha.\beta + \beta.\alpha \not\models \varphi$.

At this point, we propose the following new interpretation of propositional constants in L_t^p, based on which the preorder deduced by L_t^p coincides with performance preorder: $d \triangleright n' \models n$ *iff* $n' \geq n$

Proposition 10. *Let* $s_1,\ s_2 \in \mathcal{S}$. $s_1 \sqsubseteq s_2$ *iff* $\forall \phi \in L_t^p,\ s_1 \models \phi$ *implies* $s_2 \models \phi$.

Immediately from this proposition, we have

Theorem 11. *Let* $E_1,\ E_2 \in \mathcal{E}$. $E_1 \sqsubseteq E_2$ *iff*

$$(\forall \phi \in L_t^p,\ (0 \Rightarrow E_1) \triangleright 0 \models \phi \textit{ implies } (0 \Rightarrow E_2) \triangleright 0 \models \phi).$$

4 Specifying and Verifying Performance Properties

Many temporal and modal logics have been proposed in the literature for abstract specifications of concurrent systems [10, 14, 16], since they can be used to describe system properties, such as necessity, possibility, eventuality, etc. The logics we have discussed previously (L_t and L_t^p), can be used to express local capabilities and necessities, but they can not express safety properties (something bad will never happen) or liveness properties (something good will eventually happen), which are very useful abstract descriptions of systems behaviour.

Modal μ-calculus [15] has been recognized as a powerful logic language to describe safety and liveness properties. Furthermore, techniques have been provided to verify the truth of a formula in a given process [5, 6, 20]. What we consider here is, in fact, HML extended by fixpoint operators, denoted by $\mathcal{M}_\mu$. The formulas are generated by

$$\varphi ::= \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \langle\alpha\rangle\varphi \mid n \mid Z \mid \mu Z.\varphi$$

where the constants n are interpreted as in L_t, and

$$s \models \mu Z.\phi \quad \textit{iff} \quad \exists n.\ s \models (\mu Z.\ \phi)^n$$

Here $(\mu Z.\ \phi)^n$ is used for the n-th approximations of the least fixpoint formula:

$$\begin{array}{ll} s \models (\mu Z.\ \phi)^0 & \text{always holds} \\ s \models (\mu Z.\ \phi)^{n+1} & \textit{iff } s \models \phi[Z/(\mu Z.\ \phi)^n] \end{array}$$

and $[Z/\phi]$ denotes the syntactic substitution of ϕ for the free occurrences of Z.

We will also use $[\alpha]$ and $\nu Z.\ \varphi$ as the duals of $\langle\alpha\rangle$ and $\mu Z.\ \varphi$ respectively:

$$[\alpha]\varphi = \neg\langle\alpha\rangle\neg\varphi \qquad \nu Z.\ \varphi = \neg\mu Z.\ \neg(\varphi[Z := \neg Z])$$

with $\varphi[Z := \neg Z]$ the result of replacing all free occurrences of Z in φ with $\neg Z$.

The constant formula tt is definable as $\nu Z.\ Z$, and we will use it directly for convenience. As the usual syntactic restriction on $\mu Z.\ \varphi$, we require that all free occurrences of Z in φ lie within the scope of an even number of negations.

The least fixpoint is useful for expressing *eventual* properties, while the greatest one is useful to describe *invariance*. For instance,

$$\mu Z.(\varphi \vee (\bigvee^{\alpha \in A} \langle \alpha \rangle tt \wedge \bigwedge_{\alpha \in A} [\alpha] Z))$$

expresses that property φ is eventually satisfied along any computations, while

$$\nu Z.\ (\varphi \wedge \bigwedge_{\alpha \in A} [\alpha] Z)$$

expresses that φ is satisfied by all states along all the computations.

Example 5. Note that formula $\langle \alpha_0 \rangle 10$ states that it is possible to do an α_0 action ending at time 10; $\langle \alpha_0 \rangle \bigvee_{n \leq 10} n$ states that it is possible to do an α_0 action ending before time 10. Thus, formula

$$\mu Z.((\langle \alpha_0 \rangle \bigvee^{n \leq 10} n) \vee (\bigvee^{\alpha \in A} \langle \alpha \rangle tt \wedge \bigwedge_{\alpha \in A} [\alpha] Z))$$

expresses that along all computations, it is possible to do an α_0 action at some point before time limit 10.

The verification of a particular property expressed in $\mathcal{M}_\mu$ can be done for finite state processes, even automatically [6], in polynomial time [5]. However, the number of states in the timed transition system is in fact infinite, due to the increase of the clock values.

Now we explain how to exploit existing model checkers on finite state processes, to verify performance properties in the infinite structures we generate. The observation we use is that the timed transition system TS could be cut into a finite state transition system TS' in a way such that for any s not in the states of TS', the verification for $(TS, s) \models \varphi$ can be reduced into the one for $(untimed(TS), untimed(s)) \models \mathcal{F}(\varphi)$ where $untimed(TS)$ denotes the underlying (untimed and *finite state*) transition system of TS, and $\mathcal{F}(\varphi)$ is the untimed formula of φ defined by

$$\mathcal{F}(\varphi) = \begin{cases} tt & \varphi = tt \\ \neg \mathcal{F}(\varphi') & \varphi = \neg \varphi' \\ \mathcal{F}(\varphi_1) \wedge \mathcal{F}(\varphi_2) & \varphi = \varphi_1 \wedge \varphi_2 \\ \langle \alpha \rangle \mathcal{F}(\varphi') & \varphi = \langle \alpha \rangle \varphi' \\ \neg tt & \varphi = n \\ Z & \varphi = Z \\ \mu Z.\mathcal{F}(\varphi') & \varphi = \mu Z.\varphi' \end{cases}$$

The states s having the above property are those that have the global clocks greater than the maximum time appeared in φ. This fact is stated in Theorem

13. Let $\mathcal{T}(\varphi)$ denote the maximum time appeared in φ:

$$\mathcal{T}(\varphi) = \begin{cases} 0 & \varphi = tt \\ \mathcal{T}(\varphi') & \varphi = \neg\varphi' \\ max\{\mathcal{T}(\varphi_1), \mathcal{T}(\varphi_2)\} & \varphi = \varphi_1 \wedge \varphi_2 \\ \mathcal{T}(\varphi') & \varphi = \langle\alpha\rangle\varphi' \\ n & \varphi = n \\ 0 & \varphi = Z \\ \mathcal{T}(\varphi') & \varphi = \mu Z.\varphi' \end{cases}$$

and given $TS = \langle S, A, \rightarrow, s_0\rangle$ and number n, we use $H(TS, n)$ to denote the set of states with global clocks greater than n: $H(TS, n) = \{d \triangleright n' \in S \mid n' > n\}$

Lemma 12. $\forall n \geq \mathcal{T}(\varphi),\ s \in H(TS, n) :\ (TS, s) \models \varphi$ iff $(TS, s) \models \mathcal{F}(\varphi)$

Theorem 13. $\forall s \in H(TS, \mathcal{T}(\varphi)) :$

$(TS, s) \models \varphi$ iff $(untimed(TS), untimed(s)) \models \mathcal{F}(\varphi)$.

Now we show how this theorem can be used to define a verification strategy for performance properties. We first need to define – for a given formula φ – the relevant (finite state) "prefix" $TS|_\varphi$ of the timed transition system TS, and its underlying untimed transition system. Obviously, the finite prefix should contain all those states s such that $s \notin H(TS, \mathcal{T}(\varphi))$.

Definition 14. Given timed transition system $TS = \langle S, A, \rightarrow, s_0\rangle$, and $\mathcal{M}_\mu$ formula φ, define $TS|_\varphi = \langle S' \cup Leaves(TS, \varphi), A, \rightarrow', s_0\rangle$, where

$S' = S - H(TS, \mathcal{T}(\varphi))$

$Leaves(TS, \varphi) = \{r \in H(TS, \mathcal{T}(\varphi)) \mid \exists \alpha, n, s \in S' \text{ such that } s \xrightarrow{(\alpha,n)} r\}$

$\rightarrow' = \{(s, (\alpha, n), r) \mid s \in S',\ (s, (\alpha, n), r) \in \rightarrow\}$

To make easier the definition of the verification strategy, we introduce a new notion of satisfaction relation $\models'$. Given two transition systems $TS_1 = \langle S_1, A, \rightarrow_1, s_0\rangle$ and TS_2, a set of states S where $S \subseteq S_1$, and a state $s \in S_1$, we define a satisfaction relation $\models'$ between a quadruple (TS_1, TS_2, S, s) and a formula φ, with the intuitive meaning that the verification is done in TS_1, but during the procedure, whenever it is necessary to verify the truth of a subformula φ' of φ in a state $s' \in S$, it should be changed into verifying $\mathcal{F}(\varphi')$ in TS_2.

$$\begin{array}{lll} s \in S,\ (TS_1, TS_2, S, s) \models' \varphi & \textit{iff} & (TS_2, s) \models \mathcal{F}(\varphi) \\ s \notin S,\ (TS_1, TS_2, S, s) \models' \neg\varphi & \textit{iff} & (TS_1, TS_2, S, s) \not\models' \varphi \\ s \notin S,\ (TS_1, TS_2, S, s) \models' \varphi_1 \wedge \varphi_2 & \textit{iff} & (TS_1, TS_2, S, s) \models' \varphi_1 \text{ and} \\ & & (TS_1, TS_2, S, s) \models' \varphi_2 \\ s \notin S,\ (TS_1, TS_2, S, s) \models' \langle\alpha\rangle\varphi & \textit{iff} & \exists r, n.\ s \xrightarrow{(\alpha,n)}_1 r \text{ and} \\ & & (TS_1, TS_2, S, r) \models' \varphi \\ s \notin S,\ (TS_1, TS_2, S, s) \models' \mu Z.\varphi & \textit{iff} & \exists n.\ (TS_1, TS_2, S, s) \models' (\mu Z.\ \varphi)^n \end{array}$$

The existing tools for verifying $\models$ could be easily extended to verify this $\models'$, if both TS_1 and TS_2 are finite.

As an important fact formally stated in Proposition 1(i), our new operational semantics avoids the computations that are well-caused but ill-timed. Thus, the global clock keeps increasing along the computations. This guarantees that $TS|_\varphi$ is finite, with $Leaves(TS, \varphi)$ the set of leaves, containing exactly all states reachable from S' in one step of transition. So according to the definition of $\models'$ and Theorem 13, we have

$$(TS, s_0) \models \varphi \quad \textit{iff} \quad (TS|_\varphi, \textit{untimed}(TS), Leaves(TS, \varphi), s_0) \models' \varphi$$

5 Conclusion

In this paper we have proposed a new version of timed semantics for nets of automata where actions are patient, but not too lazy. The resulting equivalence and preorder have nice mathematical properties and possess simple logic characterizations which can be easily exploited to check performance related properties.

The kinds of performance analysis we can do are related to the hypoteses on action executions we have taken. Actions can be delayed but with an upper bound given by the current global clock. This means that the system as a whole is anyway urgent, i.e., never stays idle. Two systems are equated if, considering the whole range of their executions going from the hypothesis of maximal parallelism (best urgent well-timed case) to the hypothesis of having just one processor (worst urgent well-timed case), they perform the same actions in the same amount of time. In other words, if $P \sim Q$ in our approach, then P and Q are executed urgently in equivalent ways on any kind of architecture (= number of processors) they can be actually implemented, even if the allocation of processes to processors can vary during the execution (see Example 1).

Among other approaches to time, we would like to mention [21], developing a theory of timed processes within Petri net theory. Even if in a different context, Vogler's work presents some analogies with ours, at least because both approaches assume a non-interleaving point of view. Further work will be devoted to study the relationships between the two approaches.

References

1. L. Aceto and D. Murphy. On the ill–timed but well–caused. In E. Best, editor, *CONCUR'93, LNCS 715*, pages 97–111. Springer-Verlag, 1993.
2. L. Aceto and D. Murphy. Timing and causality in process algebra. Technical Report 9/93, University of Sussex, 1993. To appear in Acta Informatica.
3. S. Arun-Kumar and M. Hennessy. An efficient preorder for processes. *Acta Informatica*, 29:737–760, 1992.
4. S. Brooks and W. Rounds. Behavioural equivalence relations induced by programming logics. In *LNCS 154*, pages 97–108. Springer-Verlag, 1983.
5. R. Cleaveland. Tableau-based model checking in the propositional μ-calculus. *Acta Informatica*, 27:725–747, 1990.

6. R. Cleaveland, J. Parrow, and B. Steffen. The concurrency workbench: A semantics-based tool for the verification of concurrent systems. *ACM Transaction on Programming Languages and Systems*, 15:36–72, 1993.
7. F. Corradini. Compositionality for processes with durational actions. In *Proc. Fifth Italian Conference on Theoretical Computer Science*, 1995.
8. F. Corradini, R. Gorrieri, and M. Roccetti. Performance preorder and competitive equivalence. Technical Report 95/01, University of Bologna, 1995.
9. F. Corradini, R. Gorrieri, and M. Roccetti. Performance preorder: Ordering processes with respect to speed. In *MFCS'95, LNCS 969*, pages 444–453, 1995.
10. E.A. Emerson and J.Y. Halpern. "Sometimes" and "Not Never" revisited: on branching time versus linear time temporal logic. *Journal of ACM*, 33(1):151–178, 1986.
11. G-L. Ferrari and U. Montanari. Dynamic matrices and the cost analysis of concurrent programs. In *AMAST'95*, 1995.
12. R. Gorrieri and M. Roccetti. Towards performance evaluation in process algebras. In *AMAST'93*, pages 289–296. Springer-Verlag, 1993.
13. R. Gorrieri, M. Roccetti, and E. Stancampiano. A theory of processes with durational actions. *Theoretical Computer Science*, 140(1):73–94, 1995.
14. M. Hennessy and R. Milner. Algebraic laws for nondeterminism and concurrency. *J. of ACM*, 32(1):137–161, 1985.
15. D. Kozen. Results on the propositional μ-calculus. *Theoretical Computer Science*, 27:333–354, 1983.
16. Z. Manna and A. Pnueli. The anchored version of the temporal framework. In J. de Bakker, P. de Roever, and G. Rozenberg, editors, *Linear Time, Branching Time and Partial Order in Logics and Models for Concurrency, LNCS 354*, pages 201–284. Springer-Verlag, 1989.
17. R. Milner. *Communication and Concurrency.* Prentice Hall International, 1989. International Series on Computer Science.
18. F. Moller and C. Tofts. Relating processes with respect to speed. In *CONCUR'91, LNCS 527*, pages 424–438, 1991.
19. R. De Nicola and V. Vaandrager. Three logics for branching bisimulation. *Journal of ACM*, 42(2):458–487, 1995.
20. C. Stirling. Modal and temporal logics. In *Handbook of Logic in Computer Science, Volume 2*, pages 477–563. Oxford University Press, 1992.
21. W. Vogler. Faster asynchronous systems. In *Proc. CONCUR'95, LNCS 962*, pages 299–312. Springer, 1995.

$$Act \quad \frac{t \geq n + f(\alpha) \text{ and } n' \leq t \leq n' + f(\alpha)}{(n \Rightarrow \alpha.p) \triangleright n' \xrightarrow{(\alpha,t)} (t \Rightarrow p) \triangleright t}$$

$$Wait \quad \frac{t \geq n + n'', \text{ and } n' \leq t \leq n' + n''}{(n \Rightarrow \text{wait } n''.p) \triangleright n' \xrightarrow{(\tau,t)} (t \Rightarrow p) \triangleright t}$$

$$Alt_1 \quad \frac{(n \Rightarrow p_1) \triangleright n' \xrightarrow{(\mu,t)} (t \Rightarrow p_1') \triangleright t}{(n \Rightarrow p_1 + p_2) \triangleright n' \xrightarrow{(\mu,t)} (t \Rightarrow p_1') \triangleright t}$$

$$Alt_2 \quad \frac{(n \Rightarrow p_2) \triangleright n' \xrightarrow{(\mu,t)} (t \Rightarrow p_2') \triangleright t}{(n \Rightarrow p_1 + p_2) \triangleright n' \xrightarrow{(\mu,t)} (t \Rightarrow p_2') \triangleright t}$$

$$Rec \quad \frac{(n \Rightarrow p[\text{rec } v.\, p/v]) \triangleright n' \xrightarrow{(\mu,t)} (t \Rightarrow p') \triangleright t}{(n \Rightarrow \text{rec } v.\, p) \triangleright n' \xrightarrow{(\mu,t)} (t \Rightarrow p') \triangleright t}$$

$$Par_1 \quad \frac{d_1 \triangleright n \xrightarrow{(\mu,t)} d_1' \triangleright t}{(d_1 \mid d_2) \triangleright n \xrightarrow{(\mu,t)} (d_1' \mid d_2) \triangleright t}$$

$$Par_2 \quad \frac{d_2 \triangleright n \xrightarrow{(\mu,t)} d_2' \triangleright t}{(d_1 \mid d_2) \triangleright n \xrightarrow{(\mu,t)} (d_1 \mid d_2') \triangleright t}$$

$$Synch \quad \frac{d_1 \triangleright n \xrightarrow{(a,t)} d_1' \triangleright t, \qquad d_2 \triangleright n \xrightarrow{(\bar{a},t)} d_2' \triangleright t}{(d_1 \mid d_2) \triangleright n \xrightarrow{(\tau,t)} (d_1' | d_2') \triangleright t}$$

$$Res \quad \frac{d \triangleright n \xrightarrow{(\mu,t)} d' \triangleright t \text{ and } \mu \notin \{\alpha, \bar{\alpha}\}}{d \backslash \{\alpha\} \triangleright n \xrightarrow{(\mu,t)} d' \backslash \{\alpha\} \triangleright t}$$

$$Rel \quad \frac{d \triangleright n \xrightarrow{(\mu,t)} d' \triangleright t}{d[\Phi] \triangleright n \xrightarrow{(\Phi(\mu),t)} d'[\Phi] \triangleright t}$$

Table 1. The Structural Rules for the Operational Semantics.