Undecidability of performance equivalence of Petri nets

Sławomir Lasota* and Marcin Poturalski

[1] Institute of Informatics, Warsaw University
sl@mimuw.edu.pl
[2] EPFL, Lausanne, Switzerland
marcin.poturalski@epfl.ch

Abstract. We investigate bisimulation equivalence on Petri nets under durational semantics. Our motivation was to verify the conjecture that in durational setting, the bisimulation equivalence checking problem becomes more tractable (which is the case, e.g., over communication-free nets). We disprove this conjecture in three of four proposed variants of durational semantics. The fourth case remains an interesting open problem.

1 Introduction

Bisimulation equivalence [15, 14] is one of the most relevant semantical equivalences of concurrent systems. One of its advantages is that it often allows for an efficient verification algorithms in settings where other approaches (see, e.g., [7]), like language equality, lead to undecidable verification problems. There is now a wide range of results about decidability and complexity of different variants of bisimulation equivalences in different classes of infinite-state systems (see e.g. [2]).

Bisimulation equivalence relates processes exhibiting the same behaviour. In this paper we investigate *performance equivalence*, a variant of bisimulation equivalence that aims at relating not only purely functional behaviour, but also effectiveness of processes. A basic assumption is that each action of a process has assigned a positive *duration*, that is amount of time (or other resource) necessary to complete this action. Performance equivalence is then a variant of bisimulation that respects amount of time (resource) requested in both processes during execution. This notion was introduced in [8] and then studied among the others in [4, 5, 1, 11, 12].

A starting point for our investigations was an observation made in [1] that the complexity of performance equivalence may be substantially lower than complexity of ordinary bisimulation equivalence. The authors of [1] investigated so called Basic Parallel Processes (BPP in short), a natural and simple fragment

* Partially supported by EU-FET-IP *Software Engineering for Service-Oriented Overlay Computers* SENSORIA.

of process algebra CCS [14] (it is expressibly equivalent to CCS without communication). BPP, when transformed to a normal form [3], is equivalent to a subclass of communication-free Petri nets. While bisimulation equivalence on BPP in normal form is PSPACE-complete [10, 18], in [1] it was shown that performance equivalence may be decided in polynomial time. An intuitive justification of this is that the latter equivalence, being more discriminating, satisfies stronger decomposition properties and hence is more tractable. Later on, it was shown that the polynomial time procedure exists for the whole BPP (not necessarily in the normal form studied in [1]) and that it coincides with distributed equivalence [12].

Performance equivalence is computationally more tractable than bisimulation equivalence on BPP, i.e., on communication-free Petri nets, hence a natural question arises: is it also more tractable in the case of general Petri nets? As bisimulation equivalence is undecidable in this case [9], the crucial question is whether performance equivalence is decidable or not. This is the main problem investigated in this paper.

However, when one tries to define the durational semantics over Petri nets, necessary to host the notion of performance equivalence, it quickly turns out that there is no unique such semantics. We made a systematic research of possible ways to define it and come up with four different variants of durational semantics. The distinction depends on the choice between global-time or local-time approach, and on the way of synchronisation (patient and impatient approach).

As a main result, we proved undecidability of performance equivalence under three of the four semantics. While the patient variants are easily undecidable, the proof for global-time impatient semantics is nontrivial and constitutes the main technical contribution of the paper. The proof builds on the method of Jančar [9]; however, substantial new insight was necessary to come over new difficulties and subtleties appearing in the durational setting.

Under the impatient local-time semantics, the question is still open. If decidable, performance equivalence would be one of very few notions of equivalence of general Petri nets exhibiting a decision procedure. This is actually the main motivation of this paper:

> **Motivation:** basing on the positive impact of durational semantics on complexity of equivalence-checking for BPP-nets, attempt to prove dedidability for general nets, thus discovering a decidable bisimulation-like equivalence of general nets.

There is a wide range of research on timed extensions of Petri nets, like time nets or timed nets [13, 16]. In most of these extensions, some timing restrictions are posed on transition, places, or arcs. However, we would like to stress that our durational setting is different from the timed ones. The principal difference is that we do not aim at modelling timed behaviour; instead, our aim is to *measure* effectiveness of processes. In particular, we allow for a local-time semantics, where the time-stamps observed during an execution of a net need not be a monotonic sequence.

Another distinguishing aspect is that the timed settings usually properly extend ordinary untimed nets, therefore decision problems become never easier, and typically harder. As a relevant example, the reachability problem, decidable for ordinary nets, becomes undecidable both for time nets and timed nets [6].

Our durational setting *does not* subsume ordinary nets. This gives hope for decidability of performance equivalence, and this stands behind the fact that the reachability problem is decidable in all four variants (in contrast to time nets and timed nets). This topic is discussed in detail in the last section. The durational setting does properly extend ordinary nets only when we allow for zero durations; see [11] for a decidability results about performance equivalence over BPP.

In Section 2 we introduce the background material and formalise the durational setting. We also mention quickly undecidability in both patient variants. Then Section 3 is devoted to the proof of undecidability under global-time impatient variant. The last section contains a brief discussion on decidability of reachability problem for durational Petri nets and some final remarks.

2 Preliminaries

Bisimulation equivalence. A labelled transition system consists of a set of states $\mathcal{S}$ and a family of binary relations $\{\xrightarrow{a}\}_{a\in\mathcal{L}}$ indexed by a labelling set $\mathcal{L}$. We will write $s_1 \xrightarrow{a} s_2$ instead of $(s_1, s_2) \in \xrightarrow{a}$.

A relation $\mathcal{R} \subseteq \mathcal{S} \times \mathcal{S}$ is a *bisimulation* if for each $(s_1, s_2) \in \mathcal{R}$ the following two conditions hold:

- if $s_1 \xrightarrow{a} t_1$ for some a, t_1 then there is some t_2 such that $s_2 \xrightarrow{a} t_2$ and $(t_1, t_2) \in \mathcal{R}$;
- if $s_2 \xrightarrow{a} t_2$ for some a, t_2 then there is some t_1 such that $s_1 \xrightarrow{a} t_1$ and $(t_1, t_2) \in \mathcal{R}$.

States s_1, s_2 are *bisimulation equivalent* (*bisimilar*), if there is a bisimulation $\mathcal{R}$ containing (s_1, s_2).

It is instructive to recall an alternative definition of bisimilarity, in a setting of games. A Bisimulation Game is played between two players `Spoiler` and `Duplicator`. For convenience of presentation, we view `Spoiler` as "him" and `Duplicator` as "her". The *positions* in the game are pairs $(s_1, s_2) \in \mathcal{S} \times \mathcal{S}$. In a position (s_1, s_2), `Spoiler` chooses $i \in \{1, 2\}$ and a transition from s_i, say $s_i \xrightarrow{a} t_i$; `Duplicator` must respond by choosing some transition with the same label a from the other element of (s_1, s_2), i.e., a transition $s_{3-i} \xrightarrow{a} t_{3-i}$. The play then continues from the position (t_1, t_2). If one of the players gets stuck (there is no appropriate transition), the other player wins. If the play continues forever, `Duplicator` wins unconditionally.

Generally speaking, a *strategy* for a player P in a game is a (partial) function which determines a concrete P-move for each sequence of so far played moves after which it is P's turn. A strategy is *winning* if player P wins each play when he/she uses the strategy. In what follows, by a strategy we always mean a *memoryless strategy*: each prescribed move depends on the current position only, not on the whole sequence of so-far played moves. By standard results, for each position (s_1, s_2), precisely one of the players has a memoryless winning strategy; moreover we have:

Proposition 1 ([19]). *Two states s_1 and s_2 are bisimilar iff* `Duplicator` *has a winning strategy in Bisimulation Game starting from position (s_1, s_2).*

Hence `Spoiler` has a winning strategy iff s_1 and s_2 are *not* bisimilar.

Labelled Petri nets. A finite multiset over a set $\mathcal{A}$ is formally a mapping M from $\mathcal{A}$ to $\mathbb{N}$, the set of natural numbers, such that $M(a) > 0$ for only a finite number of elements a. E.g., the empty multiset $\emptyset$ maps all $a \in \mathcal{A}$ to 0. We apply the usual arithmetical operations to multisets in a point-wise manner. E.g., point-wise addition is the union operation of multisets; and $M \leq M'$ means that $M(a) \leq M'(a)$ for each $a \in \mathcal{A}$. We will often write finite multisets by enumerating its elements, e.g., aab will denote the function mapping a to 2, b to 1, and all other elements of $\mathcal{A}$ to 0.

A *labelled Petri net* (*net* in short) N is given by a finite set of places $\mathcal{P}$, a finite set $\mathcal{L}$ of labels, and a finite set of transition rules of the form $X \stackrel{a}{\leadsto} Y$, where X and Y are finite nonempty multisets over $\mathcal{P}$ and $a \in \mathcal{L}$. In the sequel it will be sufficient to consider only those nets, where X and Y are always sets (i.e., multisets mapping each element to 0 or 1). This restriction corresponds to the class of pure Place/Transition labelled Petri nets.

A net naturally induces a labelled transition system. Its states are all finite multisets over $\mathcal{P}$ (traditionally called *markings*). There is a transition from marking M to M', labelled by a, if there is a transition rule $X \stackrel{a}{\leadsto} Y$ such that $M \geq X$ (point-wise) and $M' = M - X + Y$. In traditional terminology, one says that the transition rule $X \stackrel{a}{\leadsto} Y$ is *fired* at marking M; the transition rule is *fireable* in M if $M \geq X$. Note that we do not fix an *initial marking* of a net.

There is a natural operational interpretation of the induced transition system. A marking may be understood as an assignment of a number of *tokens* to each place. And each transition rule specifies the necessary condition on the number of tokens on some places. For instance, a rule $ppq \stackrel{a}{\leadsto} ps$ would mean that an a-transition is allowed assumed at least two tokens on place p and at least one on place q; as an outcome of the transition, the three tokens are removed, then one token is placed back on p and one on s.

The bisimulation equivalence problem for labelled Petri nets is defined as follows: given a net N and two markings M_1, M_2, decide whether they are bisimilar as the states in the labelled transition system induced by N. Undecidability of this problem was proved by Jančar in [9]. Note that equivalently we could formulate the problem for the (initial) markings in two distinct nets.

Bisimulation equivalence relates two states (or two systems, in general) with the same functional behaviour. We will now extend this setting to allow to compare, in addition to pure functional behaviour, also its effectiveness (performance). The idea comes from [8] and amounts to (i) assigning a *duration* to each transition; (2) respecting the durations in the bisimulation equivalence.

Durational labelled Petri nets. Following [8], we choose a discrete time domain, represented by natural numbers. From now on we will assume that each transition rule r of any labelled Petri net N has assigned a positive natural number, its *duration*, written $\mathtt{dur}(r)$. Such nets, enriched by a duration function, are called *durational nets* in the sequel. Note that we do not allow for $\mathtt{dur}(r) = 0$, as in [8] and in the following papers [4, 5, 1]. If we allowed for zero durationsm our model would trivially subsume ordinary Petri nets.

A *durational marking* is a finite multiset over $\mathcal{P} \times \mathbb{N}$; intuitively, each token in a marking has now a *time-stamp.* In an initial marking, the time-stamps of all the tokens will be usually 0. We will write $t \triangleright p$ instead of (p, t). A durational marking M may be naturally mapped to an ordinary (non-durational) marking $\mathtt{untime}(M)$ by removing all the time-stamps (untiming) but preserving the number of tokens at each place. E.g., the durational marking $M = (0 \triangleright p)(3 \triangleright p)(3 \triangleright p)(2 \triangleright q)$ would be mapped to $\mathtt{untime}(M) = pppq$.

For notational convenience we will extend the $t \triangleright \_$ notation to non-durational markings: by $t \triangleright M$ we will mean the multiset $\{t \triangleright p : p \in M\}$. E.g., $(0 \triangleright p)(3 \triangleright p)(3 \triangleright p)(2 \triangleright q)$ may be equivalently written as $(0 \triangleright p)(3 \triangleright pp)(2 \triangleright q)$.

We will define the durational semantics for Petri nets by specifying (i) when a transition rule is fireable in a durational marking M, and (ii) what is the effect of its firing. The labels in the induced transition system will be now pairs $(a, t) \in \mathcal{L} \times \mathbb{N}$. The intuitive meaning of a transition $M \xrightarrow{a,t} M'$ is that t units of time had to elapse *before* this transition became fireable. The amount of time t is always measured relative to the starting moment of an execution; in particular, the very first transition will be usually fired at $t = 0$.

Before making our durational semantics explicit, we need to introduce some notation. For a durational marking M, by $\mathtt{stamps}(M)$ denote the set (not multiset) of all time-stamps of tokens in M, and by $\mathtt{max\text{-}stamp}(M)$ the greatest time-stamp in M. Formally:

$$\mathtt{stamps}(M) = \{t : t \triangleright p \in M\} \qquad \mathtt{max\text{-}stamp}(M) = \begin{cases} \max(\mathtt{stamps}(M)) & \text{if } M \neq \emptyset \\ 0 & \text{otherwise} \end{cases}$$

We distinguish four different durational semantics, depending on whether it is patient or impatient, and whether it is global-time or local-time. In each of the four variants, we will make it precise when *a transition rule* $X \overset{a}{\rightsquigarrow} Y$ *is fireable in a durational marking* M*, at time* t *due to a submarking* $\bar{X}$. In each of the four variants, it requires that $\bar{X} \leq M$ ($\bar{X}$ is a *durational submarking* of M) and $X = \mathtt{untime}(\bar{X})$ (the rule actually applies to $\bar{X}$). In addition, the following is required, depending on the variant of semantics:

– local-time, patient semantics: $\mathtt{max\text{-}stamp}(\bar{X}) = t$.
– local-time, impatient semantics: $\mathtt{stamps}(\bar{X}) = \{t\}$.

Intuitively, in the patient semantics, a token with an "earlier" (smaller) time-stamp may "wait" for other tokens with "later" time-stamps; but this is not allowed in the impatient variant, where the tokens must agree on their time-stamps to be able to fire synchronously a transition rule.

Note that we did not assume that a transition with the smallest possible time-stamp is chosen to be fired. In the globaltime patient (impatient) semantics we additionally require this: a transition rule $X \overset{a}{\rightsquigarrow} Y$ is fireable in M, at time t due to $\bar{X}$, if it is fireable according to the local-time patient (impatient) semantics, and t is the smallest among all possible choices of a rule $X \overset{a}{\rightsquigarrow} Y$ and a submarking $\bar{X}$.

Uniformly for all four variants, whenever a transition rule $r = (X \overset{a}{\rightsquigarrow} Y)$ is fireable in M, at time t due to $\bar{X}$, then in the induced labelled transition system there is a transition $M \xrightarrow{a,t} M'$, where $M' = M - \bar{X} + \{(t + \mathtt{dur}(r)) \triangleright p : p \in Y\}$. I.e., fresh tokens are produced as specified by Y, and their time-stamps are equal to $t + \mathtt{dur}(r)$. We call t a *time label* of transition $M \xrightarrow{a,t} M'$.

When convenient, we will identify a Petri net with its induced transition system, e.g., we will speak of 'transitions' of a net. By an execution of a net we mean a finite or infinite sequence of transitions:

$$M_0 \xrightarrow{a_0,t_0} M_1 \xrightarrow{a_1,t_1} M_2 \xrightarrow{a_2,t_2} \ldots$$

Note that the initial marking M_0 is not fixed for the net. For two durational markings M, M', we say that M' is *reachable* from M if there is a finite execution that starts in M and ends in M'.

We say that two durational markings M, M' of a given durational net N are *performance equivalent* if they are bisimulation equivalent in the induced labelled transition system. The topic of this paper is undecidability of the following problem: given N, M and M', decide whether M and M' are performance equivalent.

It is an easy observation that patient semantics can faithfully simulate ordinary semantics, and therefore we have:

Proposition 2. *The problem of performance equivalence is undecidable for Petri nets under (global-time or local-time) patient semantics.*

The easy proof is by reduction from bisimulation equivalence of ordinary Petri nets. Given such a net N, we extend it by one special place p and add this place both to pre-places and post-places of any transition. In the initial marking, p is marked with a single token. Therefore during a run, place p will always have precisely one token, with the "latest" time-stamp; and this time-stamp will be always observable at the next transition. Hence, performance equivalence between so extended nets coincides with bisimulation equivalence of the original ordinary nets.

The case of impatient semantics is more difficult. We were able to show undecidability only in the global-time variant. Decidability of performance equivalence in the last, local-time impatient variant, remains still open.

3 Undecidability under global-time impatient semantics

Theorem 1. *The problem of performance equivalence is undecidable for Petri nets under global-time impatient semantics.*

The proof is by reduction from the (undecidable) halting problem of Minsky machines, and occupies the rest of this section. It is motivated by the Jančar's proof for ordinary Petri nets [9]; however, a new insight was necessary to adapt this proof to the durational setting.

A Minsky deterministic 2-counter machine $\mathbb{M}$ consists of two counters $\mathtt{c}_0, \mathtt{c}_1$ and a set of n labelled instructions

$$\begin{array}{c} \mathtt{1} : \mathtt{instr}_1 \\ \ldots \\ \mathtt{n} : \mathtt{instr}_\mathtt{n} \end{array}$$

each instruction in one of the following forms:

$$\begin{array}{ll} \mathtt{i} : \mathtt{c_b} := \mathtt{c_b} + 1;\ \mathtt{goto\ j} & \text{(increment)} \\ \mathtt{i} : \mathtt{if}\ \mathtt{c_b} = 0\ \mathtt{then\ goto\ k} & \text{(zero-test or decrement)} \\ \qquad\qquad \mathtt{else}\ \mathtt{c_b} := \mathtt{c_b} - 1;\ \mathtt{goto\ j} & \\ \mathtt{n} : \mathtt{halt} & \text{(halting instruction)} \end{array}$$

Variables i, j and k range over $\{1 \ldots n\}$ and b over $\{0, 1\}$. We say that $\mathbb{M}$ *halts* if starting from $\mathtt{instr}_1$ and $\mathtt{c}_0 = \mathtt{c}_1 = 0$, the unique run of $\mathbb{M}$ ends in instruction $\mathtt{n} : \mathtt{halt}$.

Given a Minsky machine $\mathbb{M}$, we will construct a Petri net $N_\mathbb{M}$ and two markings in such a way that the markings are performance equivalent iff $\mathbb{M}$ does not halt.

There will be places in $N_\mathbb{M}$ corresponding to particular instructions of $\mathbb{M}$ and to particular counters. Formally, the set of places will be $\mathcal{P} = \{p_i, q_i, p'_i, q'_i\}_{i=1\ldots n} \cup \{b', b'', Z'_b, Z''_b\}_{b=0,1}$ and the labelling set $\mathcal{L} = \{i, d, z, \bar{z}, \tau_0, \tau_1, \omega\}$. Transition rules of $N_\mathbb{M}$ are defined as follows. For every increment instruction we define the transition rules:

$$(\mathtt{I}) \quad p_i \overset{i}{\rightsquigarrow} p_j b' b'' \qquad\qquad q_i \overset{i}{\rightsquigarrow} q_j b' b''$$

where $b \in \{0, 1\}$. For every 'zero-test or decrement' instruction we define the following transition rules:

$$\begin{array}{lrcl@{\qquad}rcl} (\mathtt{D}) & p_i b' b'' & \overset{d}{\rightsquigarrow} & p_j & q_i b' b'' & \overset{d}{\rightsquigarrow} & q_j \\ (\mathtt{Z}) & p_i & \overset{z}{\rightsquigarrow} & p'_i Z'_b Z''_b & q_i & \overset{z}{\rightsquigarrow} & q'_i Z'_b Z''_b \\ (\mathtt{Z_I}) & p'_i Z'_b Z''_b & \overset{\bar{z}}{\rightsquigarrow} & p_k & q'_i Z'_b Z''_b & \overset{\bar{z}}{\rightsquigarrow} & q_k \\ (\mathtt{Z_{II}}) & p'_i b'' Z'_b & \overset{\bar{z}}{\rightsquigarrow} & p_k & q'_i b'' Z'_b & \overset{\bar{z}}{\rightsquigarrow} & q_k \\ (\mathtt{Z_{III}}) & p'_i b'' Z''_b & \overset{\bar{z}}{\rightsquigarrow} & q_k & q'_i b'' Z''_b & \overset{\bar{z}}{\rightsquigarrow} & p_k \end{array}$$

And for the halting instruction we define:

$$(\mathtt{O}) \quad p_n \overset{\omega}{\rightsquigarrow} \emptyset$$

Finally, we define additional transition rules:

$$\begin{array}{llll}
(\mathtt{T_I}) & b'b'' & \overset{\tau_b}{\rightsquigarrow} & b'b'' \\
(\mathtt{T_{II}}) & b'Z''_b & \overset{\tau_b}{\rightsquigarrow} & b'b'' \\
(\mathtt{T_{III}}) & b'Z'_b & \overset{\tau_b}{\rightsquigarrow} & b'b''
\end{array}$$

and set the duration of every transition rule to 1. Note that each of the rules, except ($\mathtt{0}$), appears actually in two instances, for b equal to 0 or 1. We hope that it will be always clear from the context which of the two instances is considered.

As in the proof of Jančar [9], we will show that machine $\mathbb{M}$ halts if and only if two (singleton) durational markings $0 \triangleright p_1$ and $0 \triangleright q_1$ are not performance equivalent. By inspecting the rules one observes an invariant: every marking reachable from any of the two markings contains exactly one token on places in $\{p_i, p'_i, q_i, q'_i\}_{i=1\ldots n}$. In the following, a marking of one of the forms (after untiming):

$$p_i(0'0'')^x(1'1'')^y, \;\; p'_i(0'0'')^x(1'1'')^y, \;\; q_i(0'0'')^x(1'1'')^y \quad \text{or} \quad q'_i(0'0'')^x(1'1'')^y$$

will be used to represent the machine $\mathbb{M}$ being in a state that enables to execute instruction $\mathtt{instr_i}$, with the counter values $\mathtt{c_0} = x$ and $\mathtt{c_1} = y$. Note that not all reachable markings are of this forms, but every marking has a maximal sub-marking of this form, and it is this sub-marking that we use to determine the state of machine $\mathbb{M}$.

According to the global-time semantics, a transition rule may be fired with time label $t \in \mathbb{N}$ only when no transition rule is fireable with time label smaller than t. Consider now any execution (a sequence of transitions) of $N_{\mathbb{M}}$ starting from $0 \triangleright p_1$ or $0 \triangleright q_1$, and the first transition $M \xrightarrow{a,t} M'$ with time label t during this execution. All the tokens in M with time-stamp smaller than t are therefore not able to engage in a transition from M. We say these tokens are *dead* in M; formally, a token is dead in a marking M if some transition is fireable in M with time label bigger than the time-stamp of this token. Note that a token which is dead in some M, remains dead in each marking reachable from M, since the semantics is impatient.

We make the following observation:

Claim. For every $t \in \mathbb{N}$, every execution of $N_{\mathbb{M}}$ from $0 \triangleright p_1$ or $0 \triangleright q_1$ contains a marking such that the time-stamps of all non-dead tokens are equal to t (such a marking is called *t-equimarking*).

Indeed, if a transition with time label t appears in the execution, then the source marking M of the first such transition satisfies the condition (M contains no tokens with time-stamp larger than t as the duration of each transition rule is 1). Or no transition with time label t appears at all, which implies that the execution is finite and ends in a deadlock marking, in which all tokens are dead and have time-stamps less or equal t. It clearly satisfies the condition of the claim.

All transitions executed between two consecutive equimarkings (t- and (t+1)-equimarking, for some t) we call a *large step*.

Now we are prepared for the reduction. As the first case, assume that machine $\mathbb{M}$ halts; we will show a winning strategy for `Spoiler`. This strategy corresponds to the actual execution of $\mathbb{M}$, hence we call it a *correct simulation*. An execution of each instruction of the machine is represented (simulated) by one or two large steps.

A position (M, M') in a play is *conforming* if M and M' are identical, except that one of them may have a token $t \triangleright p_i$ and the other $t \triangleright q_i$, or one of M, M' may have $t \triangleright p'_i$ while the other $t \triangleright q'_i$. The starting position $(0 \triangleright p_1, 0 \triangleright q_1)$ is conforming. Consider a conforming pair of t-equimarkings:

$$t \triangleright p_i (0'0'')^x (1'1'')^y \qquad \Big| \qquad t \triangleright q_i (0'0'')^x (1'1'')^y$$

We write $(0'0'')^x$ for x copies of $0'0''$. As mentioned above, when we write $t \triangleright p_i (0'0'')^x (1'1'')^y$ we mean that time-stamps of all tokens are t. In the sequel, to succinctly write markings containing tokens with different time-stamps, we will use a union operation on multisets, denoted by $\|$. We will show how `Spoiler` can enforce a conforming pair of $(t+1)$-equimarkings during the play, at the same time simulating the execution of $\mathbb{M}$. Assume, without loss of generality, that $\mathtt{instr_i}$ acts on counter $\mathtt{c}_0$. We consider all possible types of instruction $\mathtt{instr_i}$. In each case, `Spoiler` plays on the left-hand side and `Duplicator` is constantly forced to copy `Spoiler`'s moves on the other side.

(increment) `Spoiler` executes transition (`I`), to which `Duplicator` must respond with (`I`). Next, `Spoiler` fires $x+y$ times transition rule ($\mathtt{T_I}$) (first all τ_0, then all τ_1); for future reference, we call such a sequence of ($\mathtt{T_I}$) transitions a default *step completion*. During the completion, the responses of `Duplicator` are uniquely determined, as transitions rules ($\mathtt{T_{II}}$) and ($\mathtt{T_{III}}$) are not fireable.

$$\begin{array}{c|c}
t \triangleright p_i (0'0'')^x (1'1'')^y & t \triangleright q_i (0'0'')^x (1'1'')^y \\
\big\downarrow^{(\mathtt{I})} & \big\downarrow^{(\mathtt{I})} \\
(t{+}1) \triangleright p_j (0'0'') \,\|\, t \triangleright (0'0'')^x (1'1'')^y & (t{+}1) \triangleright q_j (0'0'') \,\|\, t \triangleright (0'0'')^x (1'1'')^y \\
\big\downarrow^{(\mathtt{T_I})^x} & \big\downarrow^{(\mathtt{T_I})^x} \\
(t{+}1) \triangleright p_j (0'0'')^{x+1} \,\|\, t \triangleright (1'1'')^y & (t{+}1) \triangleright q_j (0'0'')^{x+1} \,\|\, t \triangleright (1'1'')^y \\
\big\downarrow^{(\mathtt{T_I})^y} & \big\downarrow^{(\mathtt{T_I})^y} \\
(t{+}1) \triangleright p_j (0'0'')^{x+1} (1'1'')^y & (t{+}1) \triangleright q_j (0'0'')^{x+1} (1'1'')^y
\end{array}$$

For convenience, we decorate each transition by the identifier of the relevant transition rule, in place of its label.

Both the resulting markings are $(t+1)$-equimarkings which completes the simulation of instruction $\mathtt{instr_i}$. They represent the machine $\mathbb{M}$ in a state that enables to execute instruction $\mathtt{instr_j}$, and the counters values are as expected.

(zero-test or decrement) In this case, the behaviour of `Spoiler` depends on the value of counter $\mathtt{c}_0$, i.e. the value of x.

If $x = 0$, `Spoiler` executes transition $(\mathtt{Z})$, to which `Duplicator` must respond with $(\mathtt{Z})$. Then `Spoiler` completes the step in the default manner.

$$
\begin{array}{c|c}
t \triangleright p_i(1'1'')^y & t \triangleright q_i(1'1'')^y \\
\downarrow^{(\mathtt{Z})} & \downarrow^{(\mathtt{Z})} \\
(t{+}1) \triangleright p'_i(Z'_0 Z''_0) \parallel t \triangleright (1'1'')^y & (t{+}1) \triangleright q'_i(Z'_0 Z''_0) \parallel t \triangleright (1'1'')^y \\
\downarrow^{(\mathtt{T}_{\mathtt{I}})^y} & \downarrow^{(\mathtt{T}_{\mathtt{I}})^y} \\
(t{+}1) \triangleright p'_i(Z'_0 Z''_0)(1'1'')^y & (t{+}1) \triangleright q'_i(Z'_0 Z''_0)(1'1'')^y
\end{array}
$$

Then `Spoiler` starts the next large step with transition $(\mathtt{Z}_{\mathtt{I}})$. As there are no $0''$ tokens in the marking, transitions $(\mathtt{Z}_{\mathtt{II}})$ and $(\mathtt{Z}_{\mathtt{III}})$ are not fireable and `Duplicator` must respond with $(\mathtt{Z}_{\mathtt{I}})$. Next, `Spoiler` completes the step in the default manner, reaching $(t{+}2)$-equimarkings, thus finalising the simulation of instruction $\mathtt{instr}_{\mathtt{i}}$ as expected.

$$
\begin{array}{c|c}
(t{+}1) \triangleright p'_i(Z'_0 Z''_0)(1'1'')^y & (t{+}1) \triangleright q'_i(Z'_0 Z''_0)(1'1'')^y \\
\downarrow^{(\mathtt{Z}_{\mathtt{I}})} & \downarrow^{(\mathtt{Z}_{\mathtt{I}})} \\
(t{+}2) \triangleright p_k \parallel (t{+}1) \triangleright (1'1'')^y & (t{+}2) \triangleright q_k \parallel (t{+}1) \triangleright (1'1'')^y \\
\downarrow^{(\mathtt{T}_{\mathtt{I}})^y} & \downarrow^{(\mathtt{T}_{\mathtt{I}})^y} \\
(t{+}2) \triangleright p_k(1'1'')^y & (t{+}2) \triangleright q_k(1'1'')^y
\end{array}
$$

In the other case, when $x > 0$, `Spoiler` start with transition $(\mathtt{D})$ and completes the step in the default manner.

$$
\begin{array}{c|c}
t \triangleright p_i(0'0'')^x(1'1'')^y & t \triangleright q_i(0'0'')^x(1'1'')^y \\
\downarrow^{(\mathtt{D})} & \downarrow^{(\mathtt{D})} \\
(t{+}1) \triangleright p_j \parallel t \triangleright (0'0'')^{x-1}(1'1'')^y & (t{+}1) \triangleright q_j \parallel t \triangleright (0'0'')^{x-1}(1'1'')^y \\
\downarrow^{(\mathtt{T}_{\mathtt{I}})^{x+y-1}} & \downarrow^{(\mathtt{T}_{\mathtt{I}})^{x+y-1}} \\
(t{+}1) \triangleright p_j(0'0'')^{x-1}(1'1'')^y & (t{+}1) \triangleright q_j(0'0'')^{x-1}(1'1'')^y
\end{array}
$$

In both cases the final position is a conforming pair of equimarkings.

(halting instruction) Player `Spoiler` executes transition $(\mathtt{O})$; `Duplicator` has no response and losses the game.

As the second case, assume that machine $\mathbb{M}$ does not halt; we will show a wining strategy for `Duplicator`. As long as `Spoiler` plays a correct simulation, `Duplicator`'s behaviour is determined. However, a position enabling firing of transition rule $(\mathtt{O})$ is never reached, and the game never ends – `Duplicator` wins.

`Spoiler` can diverge from the correct simulation (*cheat*) in two ways: *insignificant* or *significant.* The former means that the equimarkings are the same as in the correct simulation. This kind of cheating involves shuffling the order of transitions within one large step and changing the side on which `Spoiler` is playing. With such behaviour, he still essentially simulates the execution of $\mathbb{M}$: he will never fire transition rule (`O`), thus `Duplicator` wins.

`Spoiler` can also cheat in a significant manner, simulating a positive zero test, when it should decrease the counter. However, `Duplicator` can respond to such behaviour in a way which will make the next equimarkings identical, modulo dead tokens. Afterwards, `Duplicator` can copy `Spoiler`'s behaviour exactly, and hence win the game.

In what follows, we give an exhaustive overview of possible `Spoiler`'s cheating behaviour, and show how `Duplicator` can respond to them to ensure her win. As before, we start with a conforming equimarking position and we assume that instruction $\texttt{instr}_\texttt{i}$ uses counter $\texttt{c}_0$.

$$t \triangleright p_i(0'0'')^x(1'1'')^y \qquad \Big| \qquad t \triangleright q_i(0'0'')^x(1'1'')^y$$

We consider below all possible types of instruction $\texttt{instr}_\texttt{i}$. In each case we assume that the first move of `Spoiler` is on the left-hand side.

(increment) `Spoiler` can, in any order, execute transitions (`I`) and ($\texttt{T}_\texttt{I}$), which `Duplicator` must (and can) copy. The equimarkings reached in the end cannot be different then those appearing in the correct simulation. It is hence an insignificant cheating.

(zero-test or decrement) If counter $\texttt{c}_0$ used by the instruction is equal to 0, `Spoiler` can only cheat insignificantly. During the first large step involved in the simulation, `Spoiler` can choose an arbitrary ordering of (`Z`) and ($\texttt{T}_\texttt{I}$) transitions, as it does not influence the resulting equimarkings:

$$(t{+}1) \triangleright p_i'(Z_0'Z_0'')(1'1'')^y \qquad \Big| \qquad (t{+}1) \triangleright q_i'(Z_0'Z_0'')(1'1'')^y$$

In the second large step, `Spoiler`'s choice is again limited to shuffling of ($\texttt{Z}_\texttt{I}$) and ($\texttt{T}_\texttt{I}$) transitions, as none of the ($\texttt{Z}_\texttt{II}$), ($\texttt{Z}_\texttt{III}$), ($\texttt{T}_\texttt{II}$) or ($\texttt{T}_\texttt{III}$) transition rules is fireable. It has no influence on the resulting equimarkings, which completes the simulation of $\texttt{instr}_\texttt{i}$:

$$(t{+}2) \triangleright p_k(1'1'')^y \qquad \Big| \qquad (t{+}2) \triangleright q_k(1'1'')^y$$

Now assume that $\texttt{c}_0$ is non-zero, $x > 0$. `Spoiler` can cheat insignificantly, if he decides to decrease the counter. As before, he will be limited to shuffling around the (`D`) and ($\texttt{T}_\texttt{I}$) transitions, with no influence on the resulting equimarkings:

$$(t{+}1) \triangleright p_j(0'0'')^{x-1}(1'1'')^y \qquad \Big| \qquad (t{+}1) \triangleright q_j(0'0'')^{x-1}(1'1'')^y$$

In all the cases above, `Duplicator` was forced to copy the moves of `Spoiler`. A more interesting case is when `Spoiler` decides to cheat significantly, and executes transition (`Z`) despite that $x > 0$. `Duplicator` is forced to respond with (`Z`). Then, after completion of the step, irrespectively of the shuffling of the ($\texttt{T}_\texttt{I}$)

and (Z) transitions (no other transition rules are fireable), the equimarkings will have the from:

$$(t{+}1) \triangleright p_i'(Z_0'Z_0'')(0'0'')^x(1'1'')^y \qquad \Big| \qquad (t{+}1) \triangleright q_i'(Z_0'Z_0'')(0'0'')^x(1'1'')^y$$

Note that the only fireable τ_1-labeled transition rule is ($\mathtt{T_I}$), activated by tokens $(t{+}1) \triangleright 1'$ and $(t{+}1) \triangleright 1''$ (as there are no Z_1' or Z_1'' tokens). It does not matter for the next $(t+2)$-equimarkings how firings of ($\mathtt{T_I}$) are interleaved with the other transitions. Hence, we omit it in the presentation below and proceed as if we were in the position$^{(\dagger)}$:

$$(t{+}1) \triangleright p_i'(Z_0'Z_0'')(0'0'')^x \qquad \Big| \qquad (t{+}1) \triangleright q_i'(Z_0'Z_0'')(0'0'')^x$$

In all of the cases below (except for one, which is explained separately), `Duplicator` will respond in such a way that the resulting equimarkings are identical. From that point on she may copy exactly the `Spoiler`'s moves and hence win the game.

1. If `Spoiler` executes ($\mathtt{Z_I}$), `Duplicator` responds with ($\mathtt{Z_{III}}$). The step can be completed only by x executions of the ($\mathtt{T_I}$) transition on the left and $x{-}1$ executions of ($\mathtt{T_I}$) and one execution of ($\mathtt{T_{III}}$) on the right. These transitions have identical labels, hence it is irrelevant in what order they are executed and on which side `Spoiler` is playing.

$$\begin{array}{c|c} (t{+}1) \triangleright p_i'(Z_0'Z_0'')(0'0'')^x & (t{+}1) \triangleright q_i'(Z_0'Z_0'')(0'0'')^x \\ \downarrow^{(\mathtt{Z_I})} & \downarrow^{(\mathtt{Z_{III}})} \\ (t{+}2) \triangleright p_k \parallel (t{+}1) \triangleright (0'0'')^x & (t{+}2) \triangleright p_k \parallel (t{+}1) \triangleright (0'Z_0')(0'0'')^{x-1} \\ \downarrow^{(\mathtt{T_I})^x} & \downarrow^{(\mathtt{T_I})^u(\mathtt{T_{III}})(\mathtt{T_I})^{x-u-1}} \\ t+2 \triangleright p_k(0'0'')^x & t+2 \triangleright p_k(0'0'')^x \end{array}$$

2. If `Spoiler` executes transition ($\mathtt{Z_{II}}$), `Duplicator` responds with ($\mathtt{Z_{III}}$). Completing the step involves $x{-}1$ firings of ($\mathtt{T_I}$) and one firing of ($\mathtt{T_{II}}$) on the left, and $x{-}1$ firings of ($\mathtt{T_I}$) and one firing of ($\mathtt{T_{III}}$) on the right. As above, the `Spoiler`'s choices at this stage have no influence on the resulting equimarkings.

$$\begin{array}{c|c} (t{+}1) \triangleright p_i'(Z_0'Z_0'')(0'0'')^x & (t{+}1) \triangleright q_i'(Z_0'Z_0'')(0'0'')^x \\ \downarrow^{(\mathtt{Z_{II}})} & \downarrow^{(\mathtt{Z_{III}})} \\ (t{+}2) \triangleright p_k \parallel (t{+}1) \triangleright (0'Z_0'')(0'0'')^{x-1} & (t{+}2) \triangleright p_k \parallel (t{+}1) \triangleright (0'Z_0')(0'0'')^{x-1} \\ \downarrow^{(\mathtt{T_I})^u(\mathtt{T_{II}})(\mathtt{T_I})^{x-u-1}} & \downarrow^{(\mathtt{T_I})^u(\mathtt{T_{III}})(\mathtt{T_I})^{x-u-1}} \\ (t{+}2) \triangleright p_k(0'0'')^x & (t{+}2) \triangleright p_k(0'0'')^x \end{array}$$

3. If `Spoiler` executes transition ($\mathtt{Z_{III}}$), `Duplicator` responds with ($\mathtt{Z_{II}}$). This case is similar to the one above.
4. If `Spoiler` executes transition ($\mathtt{T_I}$), `Duplicator` responds depending on the value of x. If $x > 1$, she executes ($\mathtt{T_I}$). Notice that the tokens $(t{+}2) \triangleright (0'0'')$

have no influence on the $(t{+}1)$-time-stamped tokens. Hence `Duplicator` can continue as if the configuration was $^{(\dagger)}$, but with one less $0'$ and $0''$ tokens.

$(t{+}1) \triangleright p_i'(Z_0'Z_0'')(0'0'')^x$	$(t{+}1) \triangleright q_i'(Z_0'Z_0'')(0'0'')^x$
$\downarrow^{(\mathtt{T_I})}$	$\downarrow^{(\mathtt{T_I})}$
$(t{+}1) \triangleright p_i'(Z_0'Z_0'')(0'0'')^{x-1} \parallel (t{+}2) \triangleright (0'0'')$	$(t{+}1) \triangleright q_i'(Z_0'Z_0'')(0'0'')^{x-1} \parallel (t{+}2) \triangleright (0'0'')$

If $x = 1$, `Duplicator` executes ($\mathtt{T_{III}}$). Completing the step involves firing of ($\mathtt{Z_I}$) on the left and ($\mathtt{Z_{III}}$) on the right.

$(t{+}1) \triangleright p_i'(Z_0'Z_0'')(0'0'')$	$(t{+}1) \triangleright q_i'(Z_0'Z_0'')(0'0'')$
$\downarrow^{(\mathtt{T_I})}$	$\downarrow^{(\mathtt{T_{III}})}$
$(t{+}1) \triangleright p_i'(Z_0'Z_0'') \parallel (t{+}2) \triangleright (0'0'')$	$(t{+}1) \triangleright q_i'(0''Z_0'') \parallel (t{+}2) \triangleright (0'0'')$
$\downarrow^{(\mathtt{Z_I})}$	$\downarrow^{(\mathtt{Z_{III}})}$
$(t{+}2) \triangleright p_k(0'0'')$	$(t{+}2) \triangleright p_k(0'0'')$

5. If `Spoiler` executes ($\mathtt{T_{II}}$), `Duplicator` responds with ($\mathtt{T_{III}}$).

$(t{+}1) \triangleright p_i'(Z_0'Z_0'')(0'0'')^x$	$(t{+}1) \triangleright q_i'(Z_0'Z_0'')(0'0'')^x$
$\downarrow^{(\mathtt{T_{II}})}$	$\downarrow^{(\mathtt{T_{III}})}$
$(t{+}1) \triangleright p_i'(0''Z_0')(0'0'')^{x-1} \parallel (t{+}2) \triangleright (0'0'')$	$(t{+}1) \triangleright q_i'(0''Z_0'')(0'0'')^{x-1} \parallel (t{+}2) \triangleright (0'0'')$

Now the play can continue in one of two ways, depending on the behaviour of `Spoiler`. Before it is decided, as long as `Spoiler` executes ($\mathtt{T_I}$), `Duplicator` responds with ($\mathtt{T_I}$):

$(t{+}1) \triangleright p_i'(0''Z_0')(0'0'')^{x-1} \parallel (t{+}2) \triangleright (0'0'')$	$(t{+}1) \triangleright q_i'(0''Z_0'')(0'0'')^{x-1} \parallel (t{+}2) \triangleright (0'0'')$
$\downarrow^{(\mathtt{T_I})^u}$	$\downarrow^{(\mathtt{T_I})^u}$
$(t{+}1) \triangleright p_i'(0''Z_0')(0'0'')^{x-u-1} \parallel (t{+}2) \triangleright (0'0'')^{u+1}$	$(t{+}1) \triangleright q_i'(0''Z_0'')(0'0'')^{x-u-1} \parallel (t{+}2) \triangleright (0'0'')^{u+1}$

At some point `Spoiler` must decide between executing, on the left-hand side, of ($\mathtt{Z_{II}}$) (equivalently, ($\mathtt{Z_{III}}$) on the right) or ($\mathtt{T_{III}}$) (equivalently, ($\mathtt{T_{II}}$) on the right). In the first case, `Duplicator` responds as follows:

$(t{+}1) \triangleright p_i'(0''Z_0')(0'0'')^{x-u-1} \parallel (t{+}2) \triangleright (0'0'')^{u+1}$	$(t{+}1) \triangleright q_i'(0''Z_0'')(0'0'')^{x-u-1} \parallel (t{+}2) \triangleright (0'0'')^{u+1}$
$\downarrow^{(\mathtt{Z_{II}})}$	$\downarrow^{(\mathtt{Z_{III}})}$
$(t{+}2) \triangleright p_k(0'0'')^{u+1} \parallel (t{+}1) \triangleright (0'0'')^{x-u-1}$	$(t{+}2) \triangleright p_k(0'0'')^{u+1} \parallel (t{+}1) \triangleright (0'0'')^{x-u-1}$
$\downarrow^{(\mathtt{T_I})^{x-u-1}}$	$\downarrow^{(\mathtt{T_I})^{x-u-1}}$
$(t{+}2) \triangleright p_k(0'0'')^x$	$(t{+}2) \triangleright p_k(0'0'')^x$

whereas in the latter case:

$(t{+}1) \rhd p'_i(0''Z'_0)(0'0'')^{x-u-1} \parallel (t{+}2) \rhd (0'0'')^{u+1}$	$(t{+}1) \rhd q'_i(0''Z''_0)(0'0'')^{x-u-1} \parallel (t{+}2) \rhd (0'0'')^{u+1}$
$\downarrow^{(\mathtt{T_{III}})}$	$\downarrow^{(\mathtt{T_{II}})}$
$(t{+}1) \rhd p'_i0''0''(0'0'')^{x-u-2} \parallel (t{+}2) \rhd (0'0'')^{u+2}$	$(t{+}1) \rhd q'_i0''0''(0'0'')^{x-u-2} \parallel (t{+}2) \rhd (0'0'')^{u+2}$
$\downarrow^{(\mathtt{T_I})^{x-u-2}}$	$\downarrow^{(\mathtt{T_I})^{x-u-2}}$
$(t{+}1) \rhd p'_i0''0'' \parallel (t{+}2) \rhd (0'0'')^x$	$(t{+}1) \rhd q'_i0''0'' \parallel (t{+}2) \rhd (0'0'')^x$

This is the only case so far in which the resulting markings are not identical. However, the $(t{+}1)$-time-stamped tokens on both sides are dead, whereas the $(t{+}2)$-time-stamped ones are identical. Hence, `Duplicator` can still copy in future all the transitions that `Spoiler` makes, even if it the configurations do not represent a proper machine state anymore.

6. If `Spoiler` executes $(\mathtt{T_{III}})$, `Duplicator` responds with $(\mathtt{T_{II}})$. What follows is similar to the previous point.

(halting instruction) Simulation of instruction $\mathtt{n} : \mathtt{halt}$ may happen only if `Spoiler` have cheated earlier in a significant manner. As we have shown above, in such a case the markings on both sides are identical and `Duplicator` still wins.

We have shown that the initial markings $0 \rhd p_1$ and $0 \rhd q_1$ are performance equivalent if and only if machine $\mathbb{M}$ does not halt. Hence performance equivalence is undecidable.

4 Concluding remarks

We have investigated Petri nets under four different durational semantics, and the corresponding variant of bisimulation equivalence, called performance equivalence. We have proved that in three of the four cases, performance equivalence is undecidable. Hence, unfortunately, our results do not confirm a conjecture that performance equivalence might be easier to decide than classical bisimulation equivalence. However in the fourth variant, i.e., local-time impatient semantics, we were able neither to confirm nor to falsify the conjecture. There is hence still a hope that this last variant might admit an effective decision procedure and we conjecture that this is really a case.

As an encouraging observation we mention a partial decomposition property which holds in case of local-time impatient semantics. Consider a pair of performance equivalent markings, $t \rhd M$ and $t \rhd M'$, and some transition of one of them $t \rhd M \xrightarrow{a,t} t \rhd M_1 \parallel (t{+}1) \rhd M_2$, assuming $\mathtt{dur}(a) = 1$, which may be answered by some $t \rhd M \xrightarrow{a,t} t \rhd M'_1 \parallel (t{+}1) \rhd M'_2$. An easy observation is that $(t{+}1) \rhd M_2$ and $(t{+}1) \rhd M'_2$ must be necessarily equivalent, as Spoiler may decide to use only time labels greater than t from this point on. More generally, if M and M' are equivalent, than for any t, their submarkings containing tokens with time-stamps greater than t are equivalent too.

Another optimistic fact is that reachability seems to be decidable in all four variants of durational semantics. We explain this in a bit more detail now.

One possibility of defining the reachability problem is as follows: given a net N and two durational markings M_0, M, we ask whether M is reachable from M_0 in N. This problem is easily decidable, as it is sufficient to search, roughly speaking, through a finite part of the induced labelled transition system containing transitions with time label smaller than $\texttt{max-stamp}(M)$ (the other transitions must not be used if we aim at reaching M). It seems that our comparison with non-durational nets is not fair if we stick to this formulation of the problem.

A more interesting variant of reachability is when the destination marking M is non-durational. We ask now whether some marking $\bar{M}$ is reachable in N from M_0, such that $\texttt{untime}(\bar{M}) = M$. Hence, we specify only number of tokens on each place in the destination marking, and leave their time-stamps unspecified. It seems that in all the four semantic settings, this last variant of reachability is decidable. The easiest is local-time patient semantics, which does not differ significantly from non-durational semantics in the context of reachability question. All the other three variants are different from non-durational setting in that in a given reachable marking, some of the transitions may be non-fireable in the former while fireable in the latter. That is to say, our durational settings pose some additional restriction of fireability of particular transitions. Despite this difference, it seems that reachability may be solved by viewing durational Petri nets as *nets with hierarchical inhibitor arcs* [17]. In this model one allows for emptiness tests on some places, but it is assumed that all those tests (inhibitor arcs) are ordered and that they are only executed in a hierarchical manner with respect to this order. It seems that a decidability result of [17] may be used, due to a reduction from durational nets to nets with hierarchical inhibitor arcs. In addition to the performance equivalence problem which we leave open, a detailed investigation of the reachability problem is clearly an interesting continuation of the presented work.

References

1. B. Bérard, A. Labroue, Ph. Schnoebelen. *Verifying Performance Equivalence for Timed Basic Parallel Processes.* In Proceeding of the 3rd Int. Conf. Foundations of Software Science and Computation Structures (FOSSACS'2000), LNCS **1784**, pp 35-47. 2000.
2. O. Burkart, D.Caucal, F.Moller, B.Steffen. *Verification on Infinite Structures.* In J.Bergstra, A.Ponse, S.Smolka, editors, Handbook of Process Algebra, chapter 9, pp 545-623. Elsevier Science, 2001.
3. S. Christensen. *Decidability and Decomposition in Process Algebras.* PhD thesis, University of Edinburgh, Department of Computer Science, ECS-LFCS-93-278. 1993.
4. F. Corradini. *On performance congruences for process algebras.* Information and Computation **145**(2), pp 191-230, 1998.

5. F. Corradini, R. Gorrieri, M. Roccetti. *Performance preorder and competitive equivalence.* Acta Informatica **34**(11), pp 805-835, 1997.
6. D. de Frutos Escrig, F. Cuartero Gomez, V. Valero Ruiz. *On non-decidability of reachability for timed-arc Petri nets.* Proceedings of the 8th International Workshop on Petri Nets and Performance Models, 1999.
7. R.J. van Glabbeek. *The linear-branching time spectrum.* In J.C.M. Beaten and J.W. Klop, editors, CONCUR'90, LNCS **458**, pp 278-297, Amsterdam. Springer, 1990.
8. R. Gorrieri, M. Roccetti, E. Stancampiano. *A Theory of Processes with Durational Actions.* Theoretical Computer Science **140** (1), pp 73-94, 1995.
9. P. Jančar. *Undecidablilty of Bisimilarity for Petri Nets and Some Related Problems.* Theoretical Computer Science **148** (2), pp 281-301. 1995.
10. P. Jančar. *Strong bisimilarity on basic parallel processes is PSPACE-complete.* In Proceedings of the 18th Annual IEEE Symposium on Logic in Computer Science (LICS'03), pp 218-227. IEEE Computer Society Press, 2003.
11. S. Lasota. *Decidability of strong bisimilarity for timed BPP.* In Proceedings of CONCUR'02, pp 562-578, LNCS **2421**, 2002.
12. S. Lasota. *Decidability of performance equivalence for Basic Parallel Processes.* Theoretical Computer Science **360**, pp 172-192, 2006.
13. P. Merlin. *A study of the Recoverability of Communication Protocols.* PhD Thesis, Univ. of California, 1974.
14. R. Milner. *Communication and Concurrency.* Prentice Hall, 1995.
15. D.M.R. Park. *Concurrency and automata on infinite sequences.* In Proceedings of the 5th GI Conference, LNCS **104**, pp 167-183. Springer-Verlag, 1981.
16. C. Ramchamdani. *Performance evaluation of asynchronous concurrent systems by timed Petri net.* PhD Thesis, MIT, 1973.
17. K. Reinhardt *Reachability in Petri Nets with Inhibitor arcs.* A manuscript, available on `http://www-fs.informatik.uni-tuebingen.de/~reinhard/publ.html#Rei06`.
18. J. Srba. *Strong bisimilarity and regularity of basic parallel processes is PSPACE-hard.* In Proceedings of the 19th International Symposium on Theoretical Aspects of Computer Science (STACS'02), LNCS **2285**, pp 535-546. Springer-Verlag, 2002.
19. C. Stirling. *Bisimulation, Model Checking and Other Games.* Notes for Mathfit instructional meeting on games and computation, Edinburgh, 1997.