Bisimulation on Speed: A Unified Approach

Gerald Lüttgen[1] and Walter Vogler[2]

[1] Department of Computer Science, University of York, York YO10 5DD, U.K.,
luettgen@cs.york.ac.uk
[2] Institut für Informatik, Universität Augsburg, D–86135 Augsburg, Germany,
vogler@informatik.uni-augsburg.de

Abstract. Two process–algebraic approaches have been developed for comparing two bisimulation–equivalent processes with respect to speed: the one of Moller/Tofts equips actions with lower time bounds, while the one by Lüttgen/Vogler considers upper time bounds instead.
This paper sheds new light on both approaches by testifying to their close relationship. We introduce a general, intuitive concept of "faster–than", which is formalised by a notion of *amortised faster–than preorder*. When closing this preorder under all contexts, exactly the two faster–than preorders investigated by Moller/Tofts and Lüttgen/Vogler arise. For processes incorporating both lower and upper time bounds we also show that the largest precongruence contained in the amortised faster–than preorder is not a proper preorder but a timed bisimulation. In the light of this result we systematically investigate under which circumstances the amortised faster–than preorder degrades to an equivalence.

1 Introduction

Process algebras provide a popular framework for modelling and analysing the communication behaviour of asynchronous systems. Various extensions of classic process algebras, e.g., Milner's *Calculus of Communicating Systems* (CCS) [12], are also well established in the literature, including *timed process algebras*. Timed process algebras add constructs for modelling timeouts and delays of actions, and thus enable one to reason not only about the communication, or functional, behaviour of processes but also about their timing behaviour. Despite the vast literature on timed process algebra, most of which has concentrated on capturing behaviour in terms of process equivalence and refinement, there is relatively little work on relating functionally equivalent processes with respect to speed. This is surprising since designers of distributed algorithms are very interested in knowing which one out of several possible solutions to a given problem is the most time efficient one. Indeed, time efficiency is not something that can only be decided once an algorithm is implemented — often *lower* and/or *upper time bounds* on the algorithm's actions are known at design time.

Within timed process algebra, the idea of "*faster–than*" was first addressed by Moller and Tofts [14] who studied an extension of CCS, called TACS$^{\mathrm{lt}}$ in this paper, that allows for specifying lower time bounds of actions. They proposed the

MT–preorder which refines bisimulation [12] and has recently been put on firm theoretical grounds via a full–abstraction result established by us in [11]. Previously, we had also investigated an analogous approach to extending CCS with upper time bounds of actions, which resulted in the calculus TACS^{ut} and the *LV–preorder* [10]; this preorder was also justified intuitively by a full–abstraction result. That latter work complements research in various Petri–net [8, 16] and process–algebra [4] frameworks based on a testing semantics rather than a bisimulation semantics. The main shortcoming of our previous research is that the reference preorders for the two *full–abstraction results* — though similar in spirit — are quite different in detail and indeed somewhat tuned towards the desired outcomes. Also, we have not explored, and neither have others in the literature, the consequences of combining both lower and upper time bounds in a single setting.

This paper presents a unified approach to studying faster–than preorders for asynchronous processes. It unifies the previously known results on faster–than preorders in two ways. Firstly, it proposes a natural reference preorder for relating two processes with respect to speed: the *amortised faster–than preorder*. This preorder formalises the intuition that the faster process must execute each action no later than the slower process does, while both processes must be functionally equivalent in the sense of strong bisimulation [12]; here, "no later" refers to absolute time as measured from the system start, as opposed to relative time which is used in our operational semantics and describes the passing of time between actions. Although the amortised faster–than relation is more abstract than the reference preorders of [10, 11], we show that both the MT–preorder and the LV–preorder remain fully–abstract in TACS^{lt} and TACS^{ut}, respectively.

Secondly, this paper characterises the largest precongruence contained in the amortised faster–than preorder when combining the calculi TACS^{lt} and TACS^{ut}, so as to being able to specify *both* lower *and* upper time bounds of actions. This is an important open problem in the literature, and it turns out that the resulting precongruence is not a proper preorder but an equivalence relation that is a variant of *timed bisimulation* [13]. The concluding part of this paper systematically investigates under which circumstances a proper preorder is obtained, and when exactly the amortised faster–than preorder degrades to an equivalence. For example, we get a positive result as in [10] when we extend TACS^{ut} by actions that may be delayed arbitrarily long; such *lazy* actions are useful for modelling system errors that are not bound to occur within some fixed time interval.

The full–abstraction results of this paper complete the picture of faster–than preorders within bisimulation–based process algebras. On the one hand, the various published faster–than preorders can be traced back to the same notion of "faster–than", which is rooted in the concept of *amortisation*. On the other hand, the amortisation approach highlights the limits for defining a useful faster–than preorder that fully supports *compositionality*. Due to space constraints, the proofs of our results are omitted here but can be found in a technical report [9].

2 Timed Asynchronous Communicating Systems

This section presents our process algebra TACS that combines the timed process algebras TACS^{lt} [11] and TACS^{ut} [10], both of which extend Milner's CCS [12] by permitting the specification of *lower* and respectively *upper time bounds* for the execution of actions and processes. These time bounds will be used in the next sections for comparing processes with respect to speed. Syntactically, TACS includes two types of actions: *lazy* actions α and *urgent* actions $\underline{\alpha}$; the idea is that the former can idle arbitrarily, while the latter have to be performed immediately. It also includes one clock prefixing operator "$\sigma.$", called *must–clock prefix*, for specifying minimum delays and another "$\underline{\sigma}.$", called *can–clock prefix*, for specifying maximum delays. Semantically and as in CCS, an action a or $\underline{a}$ communicates with the complements $\overline{a}$ or $\underline{\overline{a}}$, irrespective of whether either action is urgent. This communication results in an urgent internal action, if both participating actions are urgent, and a lazy internal action otherwise. Moreover, TACS adopts a concept of global, discrete time that behaves as follows: process $\sigma.P$ *must wait* for *at least* one time unit before it can start executing process P (lower time bound), while process $\underline{\sigma}.P$ *can wait* for *at most* one time unit (upper time bound); thus, $\underline{\sigma}$ can be understood as a potential time step. Upper time bounds are technically enforced by the concept of *maximal progress* [7], such that time can only pass if no urgent internal computation can be performed.

Syntax. The syntax of TACS is identical to CCS, except that we include the two clock–prefixing operators and distinguish between lazy and urgent actions, as discussed above. Formally, let Λ be a countably infinite set of lazy actions not including the distinguished unobservable, *internal* action τ. With every $a \in \Lambda$ we associate a *complementary action* $\overline{a}$, and define $\overline{\Lambda} =_{\text{df}} \{\overline{a} \mid a \in \Lambda\}$. Each lazy action $a \in \Lambda$ ($\overline{a} \in \overline{\Lambda}$, τ) has an associated urgent variant, i.e., an action $\underline{a}$ ($\underline{\overline{a}}$, $\underline{\tau}$). We define $\underline{\Lambda} =_{\text{df}} \{\underline{a} \mid a \in \Lambda\}$ and $\underline{\overline{\Lambda}} =_{\text{df}} \{\underline{\overline{a}} \mid a \in \Lambda\}$, and take $\mathcal{A}$ ($\underline{\mathcal{A}}$) to denote the set $\Lambda \cup \overline{\Lambda} \cup \{\tau\}$ ($\underline{\Lambda} \cup \underline{\overline{\Lambda}} \cup \{\underline{\tau}\}$). Complementation is lifted to $\Lambda \cup \overline{\Lambda}$ ($\underline{\Lambda} \cup \underline{\overline{\Lambda}}$) by defining $\overline{\overline{a}} =_{\text{df}} a$ ($\underline{\overline{\overline{a}}} =_{\text{df}} \underline{a}$). We let $a, b, \ldots$ ($\underline{a}, \underline{b}, \ldots$) range over $\Lambda \cup \overline{\Lambda}$ ($\underline{\Lambda} \cup \underline{\overline{\Lambda}}$) and $\alpha, \beta, \ldots$ ($\underline{\alpha}, \underline{\beta}, \ldots$) over $\mathcal{A}$ ($\underline{\mathcal{A}}$). The syntax of TACS is defined as follows:

$$P \;::=\; \mathbf{0} \mid x \mid \alpha.P \mid \underline{\alpha}.P \mid \sigma.P \mid \underline{\sigma}.P \mid P + P \mid P|P \mid P \setminus L \mid P[f] \mid \mu x.P\,,$$

where x is a *variable* taken from a countably infinite set $\mathcal{V}$ of variables, $L \subseteq \mathcal{A}\backslash\{\tau\}$ is a *restriction set*, and $f : \mathcal{A} \to \mathcal{A}$ is a *finite relabelling*. A finite relabelling satisfies the properties $f(\tau) = \tau$, $f(\overline{a}) = \overline{f(a)}$, and $|\{\alpha \mid f(\alpha) \neq \alpha\}| < \infty$. The set of all terms is abbreviated by $\widehat{\mathcal{P}}$, and we define $\overline{L} =_{\text{df}} \{\overline{a} \mid a \in L\}$. We use the standard definitions for the semantic *sort* $\mathsf{sort}(P) \subseteq \Lambda \cup \overline{\Lambda}$ of some term P, *open* and *closed* terms, and *contexts* (terms with a "hole"). Due to our restriction to *finite* relabellings, sorts of terms are guaranteed to be finite so that contexts such as the one needed in the proof of Thm. 13 are well–defined. A variable is called *guarded* in a term if each occurrence of the variable is within the scope of an action– or σ–prefix. Moreover, we require for terms of the form $\mu x.P$ that x is guarded in P. Note that, since $\underline{\sigma}$ only denotes a potential time step, $\underline{\sigma}.P$ can perform the actions of P immediately, whence $\underline{\sigma}$ does not count as a guard.

We refer to closed and guarded terms as *processes*, with the set of all processes written as $\mathcal{P}$, and let $\equiv$ stand for syntactic equality.

Table 1. Operational semantics for TACS (action transitions)

Act	$\dfrac{-}{\alpha.P \xrightarrow{\alpha} P}$	uAct	$\dfrac{-}{\underline{\alpha}.P \xrightarrow{\alpha} P}$	uPre	$\dfrac{P \xrightarrow{\alpha} P'}{\underline{\sigma}.P \xrightarrow{\alpha} P'}$
Sum1	$\dfrac{P \xrightarrow{\alpha} P'}{P+Q \xrightarrow{\alpha} P'}$	Sum2	$\dfrac{Q \xrightarrow{\alpha} Q'}{P+Q \xrightarrow{\alpha} Q'}$	Rec	$\dfrac{P \xrightarrow{\alpha} P'}{\mu x.P \xrightarrow{\alpha} P'[\mu x.P/x]}$
Com1	$\dfrac{P \xrightarrow{\alpha} P'}{P\|Q \xrightarrow{\alpha} P'\|Q}$	Com2	$\dfrac{Q \xrightarrow{\alpha} Q'}{P\|Q \xrightarrow{\alpha} P\|Q'}$	Com3	$\dfrac{P \xrightarrow{a} P' \quad Q \xrightarrow{\overline{a}} Q'}{P\|Q \xrightarrow{\tau} P'\|Q'}$
Rel	$\dfrac{P \xrightarrow{\alpha} P'}{P[f] \xrightarrow{f(\alpha)} P'[f]}$	Res	$\dfrac{P \xrightarrow{\alpha} P'}{P \setminus L \xrightarrow{\alpha} P' \setminus L}\ \alpha \notin L \cup \overline{L}$		

Semantics. The *operational semantics* of a TACS term $P \in \widehat{\mathcal{P}}$ is given by a labelled transition system and an urgent action set. The labelled transition system has the form $\langle \widehat{\mathcal{P}}, \mathcal{A} \cup \{\sigma\}, \longrightarrow, P \rangle$, where $\widehat{\mathcal{P}}$ is the set of states, $\mathcal{A} \cup \{\sigma\}$ the alphabet, $\longrightarrow \subseteq \widehat{\mathcal{P}} \times (\mathcal{A} \cup \{\sigma\}) \times \widehat{\mathcal{P}}$ the transition relation, and P the start state. Transitions labelled with an action α are called *action transitions* that, like in CCS, are either internal activities or local communications in which two processes may synchronise to take a joint state change together. Transitions labelled with the clock symbol σ are called *clock transitions* representing a recurrent global synchronisation that encodes the progress of time. Note that transitions are labelled by ordinary (lazy) actions only. Urgency is dealt with in an orthogonal fashion by a notion of *urgent action set*. This is defined in Table 2 and contains exactly the urgent actions in which a term can initially engage. Note: the communication of two complementary actions results in an *urgent* silent action only if the two participating actions are urgent.

Table 2. Urgent action sets

$\mathcal{U}(\alpha.P)$	$=_{\mathrm{df}} \emptyset$	$\mathcal{U}(\underline{\alpha}.P)$	$=_{\mathrm{df}} \{\alpha\}$	$\mathcal{U}(\mathbf{0})$	$=_{\mathrm{df}} \emptyset$
$\mathcal{U}(\sigma.P)$	$=_{\mathrm{df}} \emptyset$	$\mathcal{U}(\underline{\sigma}.P)$	$=_{\mathrm{df}} \emptyset$	$\mathcal{U}(x)$	$=_{\mathrm{df}} \emptyset$
$\mathcal{U}(P \setminus L)$	$=_{\mathrm{df}} \mathcal{U}(P) \setminus (L \cup \overline{L})$	$\mathcal{U}(P[f])$	$=_{\mathrm{df}} \{f(\alpha) \mid \alpha \in \mathcal{U}(P)\}$	$\mathcal{U}(\mu x.P)$	$=_{\mathrm{df}} \mathcal{U}(P)$
$\mathcal{U}(P+Q)$	$=_{\mathrm{df}} \mathcal{U}(P) \cup \mathcal{U}(Q)$	$\mathcal{U}(P\|Q)$	$=_{\mathrm{df}} \mathcal{U}(P) \cup \mathcal{U}(Q) \cup \{\tau \mid \mathcal{U}(P) \cap \overline{\mathcal{U}(Q)} \neq \emptyset\}$		

According to our operational rules, the *action–prefix* terms $\alpha.P$ and $\underline{\alpha}.P$ may engage in action α and then behave like P. The processes $\alpha.P$ ($\alpha \in \mathcal{A}$) and $\underline{a}.P$ ($a \in \Lambda \cup \overline{\Lambda}$) may also *idle*, i.e., engage in a clock transition to themselves, as process $\mathbf{0}$ does; the rationale is that even an urgent communication action may have to wait for a communication partner. Hence, an $\underline{a}$–prefix expresses *potential* urgency which becomes actual only in a synchronisation with an urgent complementary action. The *must–clock prefix* term $\sigma.P$ can only engage in a clock transition to P; thus, σ stands for a delay of exactly one time unit, and it can be used to define lower time bounds, since P may perform further time

Table 3. Operational semantics for TACS (clock transitions)

tNil	$\dfrac{-}{\mathbf{0} \xrightarrow{\sigma} \mathbf{0}}$	tAct $\dfrac{-}{\alpha.P \xrightarrow{\sigma} \alpha.P}$	tuAct $\dfrac{-}{\underline{a}.P \xrightarrow{\sigma} \underline{a}.P}$
tPre	$\dfrac{-}{\sigma.P \xrightarrow{\sigma} P}$	tuPre $\dfrac{-}{\underline{\sigma}.P \xrightarrow{\sigma} P}$	tRec $\dfrac{P \xrightarrow{\sigma} P'}{\mu x.P \xrightarrow{\sigma} P'[\mu x.P/x]}$
tSum	$\dfrac{P \xrightarrow{\sigma} P' \quad Q \xrightarrow{\sigma} Q'}{P+Q \xrightarrow{\sigma} P'+Q'}$	tCom $\dfrac{P \xrightarrow{\sigma} P' \quad Q \xrightarrow{\sigma} Q'}{P\|Q \xrightarrow{\sigma} P'\|Q'}\ \tau \notin \mathcal{U}(P\|Q)$	
tRel	$\dfrac{P \xrightarrow{\sigma} P'}{P[f] \xrightarrow{\sigma} P'[f]}$	tRes $\dfrac{P \xrightarrow{\sigma} P'}{P \setminus L \xrightarrow{\sigma} P' \setminus L}$	

steps due to clock prefixes, lazy actions or waiting for a communication. The *can–clock prefix* term $\underline{\sigma}.P$ can additionally perform any action transition that P can engage in; in this sense, $\underline{\sigma}$ represents a delay of at most one time unit and can be used to define arbitrary upper time bounds.

The term $P|Q$ stands for the *parallel composition* of P and Q according to an interleaving semantics with synchronised communication on complementary actions resulting in the internal action τ. Time has to proceed equally on both sides of the operator. The side condition of Rule (tCom) ensures that $P|Q$ can only progress on σ, if it cannot engage in any urgent internal computation, in accordance with our notion of maximal progress. Thus, due to the urgency of the actions, $\underline{a}.P \,|\, \underline{\overline{a}}.Q$ cannot perform a time step. On the other hand, $\underline{a}.P \,|\, \underline{b}.Q$ or $\underline{a}.P \,|\, \underline{\overline{a}}.Q$ can, since communication is not possible or can at least be delayed; thus, $\underline{a}$ is urgent but also *patient*. Note that predicates within structural operational rules, such as $\tau \notin \mathcal{U}(P|Q)$ in Rule (tCom), are well understood.

The *summation operator* $+$ denotes nondeterministic choice such that $P+Q$ may behave like P or Q. Again, $P+Q$ can engage in a clock transition and delay the nondeterministic choice if and only if both P and Q can. Restriction $\setminus L$, relabelling $[f]$ and recursion $\mu x.\, P$ have the usual meaning.

The rules for action transitions are the same as for CCS, with the exception of the rules for the new can–clock prefix and for recursion; however, the latter is equivalent to the standard CCS rule over guarded terms. It is important to note that both faster–than settings previously investigated by us in [10, 11] can be found within TACS. The sub–calculus obtained when considering only lazy actions (urgent actions) and only must–clock prefixing (can–clock prefixing) is exactly the calculus $\mathrm{TACS}^{\mathrm{lt}}$ ($\mathrm{TACS}^{\mathrm{ut}}$) studied in [11] ([10]). For improving readability we also write $\mathcal{P}^{\mathrm{lt}}$ ($\mathcal{P}^{\mathrm{ut}}$) for the set of processes in $\mathrm{TACS}^{\mathrm{lt}}$ ($\mathrm{TACS}^{\mathrm{ut}}$).

The operational semantics for TACS possesses several important properties [7]. Firstly, it is *time–deterministic*, i.e., progress of time does not resolve choices. Formally, $P \xrightarrow{\sigma} P'$ and $P \xrightarrow{\sigma} P''$ implies $P' \equiv P''$, for all $P, P', P'' \in \widehat{\mathcal{P}}$, which can easily be proved by induction on the structure of P. This property is very intuitive, as only actions can resolve choices, and also technically convenient. Secondly, by our variant of *maximal progress*, a guarded term P can engage in a clock transition exactly if it cannot engage in an urgent internal

transition. Formally, $P \xrightarrow{\sigma}$ if and only if $\tau \notin \mathcal{U}(P)$, for all guarded terms P. In particular, processes in TACS$^{\mathrm{lt}}$ satisfy *laziness*: they can always engage in a clock transition. Last, but not least, we note that the sort $\mathsf{sort}(P)$ of any process P is finite. This is because we only allow *finite* relabellings.

3 Generalised Full–Abstraction Results

This section presents our unified approach to "faster–than" by introducing a very simple and intuitive preorder, the *amortised faster–than preorder*, which captures the essence of faster–than within a bisimulation–based setting, as discussed below. Using this preorder as a reference preorder, we show that the LV–preorder [10] and the MT–preorder [14] are fully–abstract within the TACS$^{\mathrm{ut}}$ and TACS$^{\mathrm{lt}}$ sub–calculi of TACS, respectively.

Definition 1 (Amortised faster–than preorder). A family $(\mathcal{R}_i)_{i\in\mathbb{N}}$ of relations over $\mathcal{P}$, indexed by natural numbers (including 0), is a *family of amortised faster–than relations* if, for all $i \in \mathbb{N}$, $\langle P, Q\rangle \in \mathcal{R}_i$, and $\alpha \in \mathcal{A}$:

1. $P \xrightarrow{\alpha} P'$ implies $\exists Q', k, l.\, Q \xrightarrow{\sigma}^{k} \xrightarrow{\alpha} \xrightarrow{\sigma}^{l} Q'$ and $\langle P', Q'\rangle \in \mathcal{R}_{i+k+l}$.
2. $Q \xrightarrow{\alpha} Q'$ implies $\exists P', k, l.\ k{+}l{\leq}i$, $P \xrightarrow{\sigma}^{k} \xrightarrow{\alpha} \xrightarrow{\sigma}^{l} P'$, and $\langle P', Q'\rangle \in \mathcal{R}_{i-k-l}$.
3. $P \xrightarrow{\sigma} P'$ implies $\exists Q', k{\geq}1{-}i.\ Q \xrightarrow{\sigma}^{k} Q'$ and $\langle P', Q'\rangle \in \mathcal{R}_{i-1+k}$.
4. $Q \xrightarrow{\sigma} Q'$ implies $\exists P',\ k{\leq}i{+}1.\ P \xrightarrow{\sigma}^{k} P'$ and $\langle P', Q'\rangle \in \mathcal{R}_{i+1-k}$.

We write $P \mathrel{\sqsupseteq}_i Q$ if $\langle P, Q\rangle \in \mathcal{R}_i$ for some family $(\mathcal{R}_i)_{i\in\mathbb{N}}$ of amortised faster–than relations, and call $\mathrel{\sqsupseteq}_0$ the *amortised faster–than preorder*.

Here, $\xrightarrow{\sigma}^{k}$ stands for k consecutive clock transitions. It is easy to show that $\mathrel{\sqsupseteq}_0$ is indeed a preorder. While reflexivity is obvious, transitivity follows immediately from the property $\mathrel{\sqsupseteq}_i \circ \mathrel{\sqsupseteq}_j \subseteq \mathrel{\sqsupseteq}_{i+j}$, for any $i, j \in \mathbb{N}$. Furthermore, $(\mathrel{\sqsupseteq}_i)_{i\in\mathbb{N}}$ is the (componentwise) largest family of amortised faster–than relations.

The above definition reflects our intuition that processes performing delays later along execution paths are faster than functionally equivalent ones that perform delays earlier; this is because the former processes are executing actions at earlier absolute times (as measured from the start of the processes). Consider, e.g., the processes $P =_{\mathrm{df}} a.b.\sigma.\sigma.c.\mathbf{0}$ and $Q =_{\mathrm{df}} \sigma.a.\sigma.b.c.\mathbf{0}$. Roughly speaking, P executes actions a, b at absolute time 0 and action c at absolute time 2. Analogously, Q executes action a at absolute time 1 and actions b, c at absolute time 2. Hence, every action in P is executed earlier than, or at the same absolute time as in Q, whence P is strictly faster than Q. This idea is formalised in the above definition as follows: Q is permitted to match an a from P by σa; the additional time step is saved as a credit by increasing the index of $\mathcal{R}$ such that P can perform this time step when needed, i.e., after its b. Thus, in Def. 1, an action or clock transition is matched by allowing the matching process fewer or more clock transitions as far as this is allowed by the available credit; the

difference in the number of clock transitions is added to or subtracted from the credit. In this sense, our definition canonically captures the idea of amortisation.

The remainder of this paper is concerned with the characterisation of the largest precongruence contained in $\precsim_0$, for various sub–calculi of TACS, in particular $\mathrm{TACS}^{\mathrm{ut}}$ and $\mathrm{TACS}^{\mathrm{lt}}$. We will also discuss below which variants of $\precsim_0$ have been used for $\mathrm{TACS}^{\mathrm{ut}}$ and $\mathrm{TACS}^{\mathrm{lt}}$ in [10, 11], and we will write $\precsim_i^{\mathrm{ut}}$ and $\precsim_i^{\mathrm{lt}}$ when restricting $\precsim_i$ to processes in $\mathrm{TACS}^{\mathrm{ut}}$ and $\mathrm{TACS}^{\mathrm{lt}}$, respectively.

3.1 The LV–Preorder is Fully Abstract in $\mathrm{TACS}^{\mathrm{ut}}$

$\mathrm{TACS}^{\mathrm{ut}}$ is the sub–calculus of TACS that emerges when restricting ourselves to urgent actions $\underline{\alpha}$ and can–clock prefixing $\underline{\sigma}$ only, i.e., disregarding lazy actions and must–clock prefixing. We start off by recalling some definitions from [10].

Definition 2 (LV–preorder [10]). A relation $\mathcal{R}$ over $\mathcal{P}^{\mathrm{ut}}$ is an *LV–relation* if, for all $\langle P, Q\rangle \in \mathcal{R}$ and $\alpha \in \mathcal{A}$:

1. $P \xrightarrow{\alpha} P'$ implies $\exists Q'.\, Q \xrightarrow{\alpha} Q'$ and $\langle P', Q'\rangle \in \mathcal{R}$.
2. $Q \xrightarrow{\alpha} Q'$ implies $\exists P'.\, P \xrightarrow{\alpha} P'$ and $\langle P', Q'\rangle \in \mathcal{R}$.
3. $P \xrightarrow{\sigma} P'$ implies $\mathcal{U}(Q) \subseteq \mathcal{U}(P)$ and $\exists Q'.\, Q \xrightarrow{\sigma} Q'$ and $\langle P', Q'\rangle \in \mathcal{R}$.

We write $P \precsim_{\mathrm{lv}} Q$ if $\langle P, Q\rangle \in \mathcal{R}$ for some LV–relation $\mathcal{R}$, and call $\precsim_{\mathrm{lv}}$ the LV–preorder.

This definition is of an elegant simplicity, since an LV–relation essentially combines bisimulation on actions with simulation on clock steps; the condition on the inclusion of urgent sets is needed to get a precongruence for parallel composition.

We also introduced in [10] an amortised variant of the LV–preorder which, in contrast to the amortised faster–than preorder of Def. 1, does not allow for leading and trailing clock transitions when matching action transitions — just as for the LV–preorder. Also, for matching clock transitions, the increase or decrease of the credit is restricted.

Definition 3 (Amortised LV–preorder [10]). A family $(\mathcal{R}_i)_{i\in\mathbb{N}}$ of relations over $\mathcal{P}^{\mathrm{ut}}$ is a *family of amortised LV–relations* if, for all $i \in \mathbb{N}$, $\langle P, Q\rangle \in \mathcal{R}_i$, and $\alpha \in \mathcal{A}$:

1. $P \xrightarrow{\alpha} P'$ implies $\exists Q'.\, Q \xrightarrow{\alpha} Q'$ and $\langle P', Q'\rangle \in \mathcal{R}_i$.
2. $Q \xrightarrow{\alpha} Q'$ implies $\exists P'.\, P \xrightarrow{\alpha} P'$ and $\langle P', Q'\rangle \in \mathcal{R}_i$.
3. $P \xrightarrow{\sigma} P'$ implies (a) $\exists Q'.\, Q \xrightarrow{\sigma} Q'$ and $\langle P', Q'\rangle \in \mathcal{R}_i$, or
   (b) $i{>}0$ and $\langle P', Q\rangle \in \mathcal{R}_{i-1}$.
4. $Q \xrightarrow{\sigma} Q'$ implies (a) $\exists P'.\, P \xrightarrow{\sigma} P'$ and $\langle P', Q'\rangle \in \mathcal{R}_i$, or
   (b) $\langle P, Q'\rangle \in \mathcal{R}_{i+1}$.

We write $P \precsim_i^{\mathrm{lv}} Q$ if $\langle P, Q\rangle \in \mathcal{R}_i$ for some family $(\mathcal{R}_i)_{i\in\mathbb{N}}$ of amortised LV–relations, and call $\precsim_0^{\mathrm{lv}}$ the *amortised LV–preorder*.

Theorem 4 (Full abstraction [10]). *The LV–preorder* $\sqsupseteq_{lv}$ *is the* largest precongruence *contained in* $\sqsupseteq_0^{lv}$.

The next theorem is the main result of this section and, because of $\sqsupseteq_0^{\mathrm{lv}} \subseteq \sqsupseteq_0^{\mathrm{ut}}$, generalises the above theorem.

Theorem 5 (Generalised full abstraction in TACS$^{\mathbf{ut}}$). *The LV–preorder* $\sqsupseteq_{lv}$ *is the* largest precongruence *contained in* $\sqsupseteq_0^{ut}$.

3.2 The MT–Preorder is Fully Abstract in TACS$^{\mathrm{lt}}$

We turn our attention to the TACS sub–calculus TACS$^{\mathrm{lt}}$ in which only lazy actions α and the must–clock prefix σ are available. Although a σ–prefix corresponds to exactly one time unit, these prefixes specify lower time bounds for actions in this fragment, since actions can always be delayed arbitrarily. We first recall the faster–than preorder introduced by Moller and Tofts in [14], to which we refer as *Moller–Tofts preorder*, or MT–preorder for short.

Definition 6 (MT–preorder [14]). A relation $\mathcal{R}$ over $\mathcal{P}^{\mathrm{lt}}$ is an *MT–relation* if, for all $\langle P, Q\rangle \in \mathcal{R}$ and $\alpha \in \mathcal{A}$:

1. $P \xrightarrow{\alpha} P'$ implies $\exists Q', k, P''.\, Q \xrightarrow{\sigma}{}^{k} \xrightarrow{\alpha} Q'$, $P' \xrightarrow{\sigma}{}^{k} P''$, and $\langle P'', Q'\rangle \in \mathcal{R}$.
2. $Q \xrightarrow{\alpha} Q'$ implies $\exists P'.\, P \xrightarrow{\alpha} P'$ and $\langle P', Q'\rangle \in \mathcal{R}$.
3. $P \xrightarrow{\sigma} P'$ implies $\exists Q'.\, Q \xrightarrow{\sigma} Q'$ and $\langle P', Q'\rangle \in \mathcal{R}$.
4. $Q \xrightarrow{\sigma} Q'$ implies $\exists P'.\, P \xrightarrow{\sigma} P'$ and $\langle P', Q'\rangle \in \mathcal{R}$.

We write $P \sqsupseteq_{\mathrm{mt}} Q$ if $\langle P, Q\rangle \in \mathcal{R}$ for some MT–relation $\mathcal{R}$, and call $\sqsupseteq_{\mathrm{mt}}$ the *MT–preorder*.

It is easy to see that $\sqsupseteq_{\mathrm{mt}}$ is indeed a preorder and that it is the largest MT–relation. We have also proved in [11] that $\sqsupseteq_{\mathrm{mt}}$ is a precongruence for all TACS$^{\mathrm{lt}}$ operators. The only difficult and non–standard part of that proof concerned compositionality regarding parallel composition and was based on the following *commutation lemma*.

Lemma 7 (Commutation lemma [11]). *Let* $P, P' \in \mathcal{P}^{lt}$ *and* $w \in (\mathcal{A} \cup \{\sigma\})^*$. *If* $P \xrightarrow{w} \xrightarrow{\sigma}{}^{k} P'$, *for* $k \in \mathbb{N}$, *then* $\exists P''.\, P \xrightarrow{\sigma}{}^{k} \xrightarrow{w} P''$ *and* $P' \sqsupseteq_{mt} P''$.

This lemma holds as well within the slightly more general setting of Sec. 5.2, in which also can–clock prefixes are allowed. We also introduced in [11] an amortised variant of the MT–preorder, which is however less abstract than the amortised faster–than preorder of Def. 1.

Definition 8 (Amortised MT–preorder [11]). A family $(\mathcal{R}_i)_{i\in\mathbb{N}}$ of relations over $\mathcal{P}^{\mathrm{lt}}$ is a *family of amortised MT–relations* if, for all $i \in \mathbb{N}$, $\langle P, Q\rangle \in \mathcal{R}_i$, and $\alpha \in \mathcal{A}$:

1. $P \xrightarrow{\alpha} P'$ implies $\exists Q', k.\, Q \xrightarrow{\sigma}{}^{k} \xrightarrow{\alpha} Q'$ and $\langle P', Q' \rangle \in \mathcal{R}_{i+k}$.
2. $Q \xrightarrow{\alpha} Q'$ implies $\exists P', k{\leq}i.\, P \xrightarrow{\sigma}{}^{k} \xrightarrow{\alpha} P'$ and $\langle P', Q' \rangle \in \mathcal{R}_{i-k}$.
3. $P \xrightarrow{\sigma} P'$ implies $\exists Q', k{\geq}0.\, k{\geq}1{-}i,\, Q \xrightarrow{\sigma}{}^{k} Q'$, and $\langle P', Q' \rangle \in \mathcal{R}_{i-1+k}$.
4. $Q \xrightarrow{\sigma} Q'$ implies $\exists P',\, k{\geq}0.\, k{\leq}i{+}1,\, P \xrightarrow{\sigma}{}^{k} P'$, and $\langle P', Q' \rangle \in \mathcal{R}_{i+1-k}$.

We write $P \sqsupseteq^{\mathrm{mt}}_{\approx_i} Q$ if $\langle P, Q \rangle \in \mathcal{R}_i$ for some family $(\mathcal{R}_i)_{i \in \mathbb{N}}$ of amortised MT–relations, and call $\sqsupseteq^{\mathrm{mt}}_{\approx_0}$ the *amortised MT–preorder*.

When comparing Defs. 8 and 1, it is obvious that $\sqsupseteq^{\mathrm{mt}}_{\approx_0} \subseteq \sqsupseteq^{\mathrm{lt}}_{\approx_0}$. While Conds. (3) and (4) coincide in Defs. 8 and 1, Conds. (1) and (2) do not allow clock transitions to trail the matching α–transition — just as it is the case in Cond. (1) in Def. 6. We recall the following full–abstraction result from [11].

Theorem 9 (Full abstraction [11]). *The MT–preorder* $\sqsupseteq_{\approx_{mt}}$ *is the* largest precongruence *contained in* $\sqsupseteq^{mt}_{\approx_0}$.

We generalise this full–abstraction result here by replacing $\sqsupseteq^{\mathrm{mt}}_{\approx_0}$ by $\sqsupseteq^{\mathrm{lt}}_{\approx_0}$.

Theorem 10 (Generalised full abstraction in TACS$^{\mathbf{lt}}$). *The MT–preorder* $\sqsupseteq_{\approx_{mt}}$ *is the* largest precongruence *contained in* $\sqsupseteq^{lt}_{\approx_0}$.

Thms. 5 and 10 testify not only to the elegance of the amortised faster–than preorder as a very intuitive faster–than preorder, but also as a unified starting point to approaching faster–than relations on processes.

4 Full Abstraction in TACS

Having identified the largest precongruences contained in the amortised faster–than preorder for the sub–calculi TACS$^{\mathrm{ut}}$ and TACS$^{\mathrm{lt}}$ of TACS, it is natural to investigate the same issue for the full calculus.

For a calculus with must–clock prefixing and urgent actions, Moller and Tofts informally argued in [14] that a precongruence relating bisimulation–equivalent processes cannot satisfy a property one would, at first sight, expect from a faster–than preorder, namely that omitting a must–clock prefix should result in a faster process. This intuition can be backed up by a more general result within our setting, which includes must–clock prefixing and urgent actions, too. Our result is not just based on a specific property; instead, we have a semantic definition of an intuitive faster–than as the coarsest precongruence refining the amortised faster–than preorder, and we will show that this precongruence degrades to a congruence, rather than a proper precongruence. This congruence turns out to be a variant of *timed bisimulation* [13].

Definition 11 (Timed bisimulation). A relation $\mathcal{R}$ over $\mathcal{P}$ is a *timed bisimulation relation* if, for all $\langle P, Q \rangle \in \mathcal{R}$ and $\alpha \in \mathcal{A}$:

1. $P \xrightarrow{\alpha} P'$ implies $\exists Q'.\, Q \xrightarrow{\alpha} Q'$ and $\langle P', Q' \rangle \in \mathcal{R}$.

2. $P \xrightarrow{\sigma} P'$ implies $\exists Q'.\, Q \xrightarrow{\sigma} Q'$ and $\langle P', Q' \rangle \in \mathcal{R}$.
3. $Q \xrightarrow{\alpha} Q'$ implies $\exists P'.\, P \xrightarrow{\alpha} P'$ and $\langle P', Q' \rangle \in \mathcal{R}$.
4. $Q \xrightarrow{\sigma} Q'$ implies $\exists P'.\, P \xrightarrow{\sigma} P'$ and $\langle P', Q' \rangle \in \mathcal{R}$.

We write $P \sim_{\mathrm{t}} Q$ if $\langle P, Q \rangle \in \mathcal{R}$ for some timed bisimulation relation $\mathcal{R}$, and call $\sim_{\mathrm{t}}$ *timed bisimulation*.

It is obvious that timed bisimulation $\sim_{\mathrm{t}}$ is an equivalence and that it refines the amortised faster–than preorder $\sqsupseteq_0$. However, $\sim_{\mathrm{t}}$ is not a congruence for TACS since it is not compositional for parallel composition. To see this, consider the processes $\underline{a}.\mathbf{0}+\underline{b}.\mathbf{0} \sim_{\mathrm{t}} \sigma.\underline{a}.\mathbf{0}+\underline{b}.\mathbf{0}$. When putting them in parallel with process $\overline{\underline{b}}.\mathbf{0}$ the relation $\sim_{\mathrm{t}}$ is no longer preserved since $(\underline{a}.\mathbf{0} + \underline{b}.\mathbf{0}) \,|\, \overline{\underline{b}}.\mathbf{0}$ can engage in an a–transition while $(\sigma.\underline{a}.\mathbf{0} + \underline{b}.\mathbf{0}) \,|\, \overline{\underline{b}}.\mathbf{0}$ cannot, as the clock transition that would enable action $\underline{a}$ is preempted by the urgent communication on $\underline{b}$. We thus have to refine timed bisimulation and take initial urgent action sets into account.

Definition 12 (Urgent timed bisimulation). A relation $\mathcal{R}$ over $\mathcal{P}$ is an *urgent timed bisimulation relation* if, for all $\langle P, Q \rangle \in \mathcal{R}$ and $\alpha \in \mathcal{A}$:

1. $P \xrightarrow{\alpha} P'$ implies $\exists Q'.\, Q \xrightarrow{\alpha} Q'$ and $\langle P', Q' \rangle \in \mathcal{R}$.
2. $P \xrightarrow{\sigma} P'$ implies $\mathcal{U}(Q) \subseteq \mathcal{U}(P)$ and $\exists Q'.\, Q \xrightarrow{\sigma} Q'$ and $\langle P', Q' \rangle \in \mathcal{R}$.
3. $Q \xrightarrow{\alpha} Q'$ implies $\exists P'.\, P \xrightarrow{\alpha} P'$ and $\langle P', Q' \rangle \in \mathcal{R}$.
4. $Q \xrightarrow{\sigma} Q'$ implies $\mathcal{U}(P) \subseteq \mathcal{U}(Q)$ and $\exists P'.\, P \xrightarrow{\sigma} P'$ and $\langle P', Q' \rangle \in \mathcal{R}$.

We write $P \simeq_{\mathrm{t}} Q$ if $\langle P, Q \rangle \in \mathcal{R}$ for some urgent timed bisimulation relation $\mathcal{R}$, and call $\simeq_{\mathrm{t}}$ *urgent timed bisimulation*.

We have used set inclusion in Conds. (2) and (4) above in analogy to Def. 2. It is important to note the following: if $P \xrightarrow{\sigma} P'$, then $Q \xrightarrow{\sigma} Q'$ by Cond. (2), so that Cond. (4) becomes applicable. Therefore, we could just as well require equality of urgent sets in Conds. (2) and (4). This equality is violated for the two processes $\underline{a}.\mathbf{0} + \underline{b}.\mathbf{0}$ and $\sigma.\underline{a}.\mathbf{0} + \underline{b}.\mathbf{0}$ considered above, although both can engage in a clock transition.

Theorem 13 (Full abstraction). *Urgent timed bisimulation $\simeq_t$ is the* largest congruence *contained in $\sim_t$.*

Theorem 14 (Full abstraction in TACS). *Urgent timed bisimulation $\simeq_t$ is the* largest (pre–)congruence *contained in $\sqsupseteq_0$.*

5 Discussion

This section investigates when exactly the amortised faster–than preorder, when closed under all contexts, collapses from a proper precongruence to a congruence. We have shown in the TACS sub–calculus with only must–clock prefixing and lazy actions (cf. Sec. 3.1) and in the sub–calculus with only can–clock prefixing and urgent actions (cf. Sec. 3.2) that indeed proper precongruences are

obtained: the MT–preorder and the LV–preorder, respectively. However, when combining both clock prefixes as well as lazy and urgent actions, then the result is a congruence: urgent timed bisimulation (cf. Sec. 4). We desire to explore where exactly this borderline lies, by characterising the largest precongruence contained in the amortised faster–than preorder for other combinations of can–/must–clock prefixes as well as urgent/lazy actions. While some of the resulting settings might not appear natural, others are clearly practically relevant, and this will be pointed out when analysing each combination in turn.

5.1 Can–Clock Prefixing and Urgent+Lazy Actions

Here we find ourselves in the sub–calculus $\mathrm{TACS}^{\mathrm{ut}}$ investigated in Sec. 3.1, where additionally lazy actions may be present. Lazy actions might be used for modelling the potential of errors: many errors in practice can occur at any moment and thus cannot be associated with maximal delays.

Corollary 15 (Full–abstraction in the can/urgent+lazy setting). *The LV–preorder $\sqsupseteq_{lv}$ is the* largest precongruence *contained in $\sqsupseteq_0$, when considering* TACS *processes with can–clock prefixes only.*

Hence, Thm. 5 of Sec. 3.1 remains valid in the presence of lazy actions; one only needs to check the proof of Thm. 5 and all the proofs of [10] on which it depends.

5.2 Must– and Can–Clock Prefixing and Lazy Actions

The setting here is the one of $\mathrm{TACS}^{\mathrm{lt}}$, where can–clock prefixes are added. This does not change the result we obtained for the $\mathrm{TACS}^{\mathrm{lt}}$ setting (cf. Thm. 10 in Sec. 3.2), when extending the definition of the MT–preorder $\sqsupseteq_{\mathrm{mt}}$ (cf. Def. 6) from processes in $\mathcal{P}^{\mathrm{lt}}$ to the class of processes considered here.

Theorem 16 (Full abstraction in the must+can/lazy setting). *The MT–preorder $\sqsupseteq_{mt}$ is the* largest precongruence *contained in $\sqsupseteq_0$, when considering* TACS *processes with lazy actions only.*

This statement can be deduced by inspecting the proofs of Sec. 3.2, i.e., the proof of Thm. 10 and the proofs of the underlying statements adopted from [11], in the presence of $\underline{\sigma}$–prefixes. The only parts that are not straightforward concern checking whether the MT–preorder $\sqsupseteq_{\mathrm{mt}}$ is also compositional for can–clock prefixes and whether the commutation lemma, Lemma 7, still holds. To do so we first need to adapt the syntactic faster–than preorder $\succ$ of [11] by adding the clause $P \succ \underline{\sigma}.P$.

Definition 17 (Syntactic Faster–Than Preorder). The relation $\succ \subseteq \widehat{\mathcal{P}} \times \widehat{\mathcal{P}}$ is defined as the smallest relation satisfying the following properties, for all $P, P', Q, Q' \in \widehat{\mathcal{P}}$.

Always:	(1) $P \succ P$	(2) (*a*) $P \succ \sigma.P$ and (*b*) $P \succ \underline{\sigma}.P$
$P' \succ P$, $Q' \succ Q$:	(3) $P'\|Q' \succ P\|Q$	(4) $P' + Q' \succ P + Q$
	(5) $P' \setminus L \succ P \setminus L$	(6) $P'[f] \succ P[f]$
$P' \succ P$, x guarded:	(7) $P'[\mu x.\, P/x] \succ \mu x.\, P$	

Lemma 18. *For any P, P', if $P \xrightarrow{\sigma} P'$ then $P' \succ P$.*

This lemma is adopted from Lemma 5(2) of the full version of [11], and its proof is by a straightforward induction on the structure of P. Also the other statements of the mentioned Lemma 5 hold under the modified syntactic faster–than preorder, in particular $P' \succ P$ implies $P' \sqsupseteq_{\mathrm{mt}} P$ for *processes* P', P in the TACS fragment we consider in this subsection. For the proof of Lemma 5 it is important that these processes satisfy the *laziness property*, i.e., each of them can perform a time step. We can now prove that the MT–preorder is compositional for can–clock prefixes, in the TACS sub–calculus that is restricted to lazy actions only.

Lemma 19. *Let P, Q be* TACS *processes with lazy actions only. Then $P \sqsupseteq_{mt} Q$ implies $\underline{\sigma}.P \sqsupseteq_{mt} \underline{\sigma}.Q$.*

Moreover, since the correctness of the commutation lemma is only based on Lemma 5 of the full version of [11], the laziness property as well as the time–determinism property, the commutation lemma obviously remains valid even in the presence of can–clock prefixing.

5.3 Can–Clock Prefixing and Lazy Actions

This combination is one that does not appear to be intuitive. If every action can delay its execution, additional potential delays specified by can–clock prefixes seem irrelevant and can be omitted (cf. Prop. 20). Further, if every delay specified by a clock prefix can indeed be omitted, then it appears that delays are not relevant at all and may thus be safely ignored (cf. Thm. 22).

Proposition 20. *$P \sim_t \underline{\sigma}.P$ for all* TACS *processes P with can–clock prefixes and lazy actions only.*

Because of the irrelevance of timed behaviour, timed bisimulation $\sim_{\mathrm{t}}$ coincides with standard bisimulation $\sim$ [12] — where clock transitions are ignored — in the setting considered in this section.

Lemma 21. *$\sim = \sim_t$ on* TACS *processes P with can–clock prefixes and lazy actions only.*

As expected, the amortised faster–than preorder, when closed under all contexts, degrades to standard bisimulation in this setting.

Theorem 22 (Full abstraction in the can/lazy setting). *Standard bisimulation $\sim$ is the* largest precongruence *contained in $\sqsupseteq_0$, when considering* TACS *processes with can–clock prefixes and lazy actions only.*

To conclude, note that Prop. 20 does not hold in the presence of must–clock prefixes; e.g., $\underline{\sigma}.\sigma.a.\mathbf{0} \xrightarrow{\sigma} \sigma.a.\mathbf{0}$ and $\sigma.a.\mathbf{0} \xrightarrow{\sigma} a.\mathbf{0}$, but obviously $\sigma.a.\mathbf{0} \not\sim a.\mathbf{0}$.

5.4 Must–Clock Prefixing and Urgent Actions, & More

For the full algebra TACS, we have shown in Sec. 4 that the largest precongruence contained in the amortised faster–than preorder is urgent timed bisimulation (cf. Thm. 14). Full TACS combines must– and can–clock prefixing with lazy and urgent actions. When leaving out either lazy actions, or can–clock prefixes, or both, the result remains valid, as can be checked by inspecting the proofs of Sec. 4. Essentially, the reason is that the context constructed within this proof uses neither lazy actions nor can–clock prefixes.

Most interesting is the case when we are left with must–clock prefixing and urgent actions only. This setting coincides with the one of Hennessy and Regan's well–known *Timed Process Language* [7], TPL, in terms of both syntax and operational semantics, when leaving out TPL's timeout operator; we refer to this calculus as TPL$^-$. It is important to note that, for TPL$^-$, urgent timed bisimulation is the same as timed bisimulation; this is because all actions are urgent, and the bisimulation conditions on actions imply that equivalent processes have the same initial (urgent) actions.

However, adding either can–clock prefixing or lazy actions to TPL$^-$ leads to a more expressive calculus than TPL$^-$. For example, the process $\underline{\sigma}.\underline{\tau}.P$ in the setting must+can–clock prefixing and urgent actions can engage in both a clock transition and a τ–transition, and the same applies to process $\tau.P$. This semantic behaviour is incompatible with the maximal–progress property in TPL$^-$, and indeed in full TPL, bearing in mind that every action is urgent.

6 Related Work

Relatively little work has been published on theories that relate processes with respect to speed. This is somewhat surprising, given the wealth of literature on timed process algebras and the importance of time efficiency in system design.

Early research on process efficiency compares untimed CCS–like terms by counting internal actions either within a testing–based [15] or a bisimulation–based [2, 3] setting. Due to interleaving, e.g., $(\tau.a.\mathbf{0} \mid \tau.\overline{a}.b.\mathbf{0}) \setminus \{a\}$ is considered to be as efficient as $\tau.\tau.\tau.b.\mathbf{0}$, whereas $(\sigma.a.\mathbf{0} \mid \sigma.\overline{a}.b.\mathbf{0}) \setminus \{a\}$ $((\underline{\sigma}.\underline{a}.\mathbf{0} \mid \underline{\sigma}.\underline{\overline{a}}.\underline{b}.\mathbf{0}) \setminus \{\underline{a}\})$ is strictly faster than $\sigma.\sigma.\tau.b.\mathbf{0}$ $(\underline{\sigma}.\underline{\sigma}.\underline{\tau}.\underline{b}.\mathbf{0})$ in our setting.

The most closely related research to ours is obviously the one by Moller and Tofts on processes equipped with lower time bounds [14] and our own on processes equipped with upper time bounds [10]. The work of Moller and Tofts has recently been revisited by us [11] and completed by adding an axiomatisation for finite processes, a full–abstraction result, and a "weak" variant of the MT–preorder that abstracts from the unobservable action τ. Our work on upper time bounds [10] features similar results for the LV–preorder. In both papers [10, 11], the chosen reference preorders for the full–abstraction results are less abstract than the amortised faster–than preorder advocated here. Although a couple of these reference preorders borrowed some idea of amortisation (cf. Defs. 3 and 8),

they were somewhat tweaked to fit the LV–preorder and the MT–preorder, respectively. Thus, Thms. 5 and 10 are indeed significant generalisations of the corresponding theorems in [10] and in [11] (cf. Thms. 4 and 9), respectively.

Most other published work on faster–than relations focuses on settings with upper time bounds and on preorders based on De Nicola and Hennessy's testing theory. Initially, research was conducted within the setting of Petri nets [16, 17], and later for the Theoretical–CSP–style process algebra PAFAS [4]. An attractive feature when adopting testing semantics is a fundamental result stating that the considered faster–than testing preorder based on continuous–time semantics coincides with the analogous testing preorder based on discrete–time semantics [17]. It remains to be seen whether a similar result holds for our bisimulation–based approach.

Last, but not least, Corradini et al. [5] introduced the *ill–timed–but–well–caused* approach for relating processes with respect to speed [1, 6]. This approach allows system components to attach local time stamps to actions. However, as a byproduct of interleaving semantics, local time stamps may decrease within action sequences exhibited by concurrent processes. These "ill–timed" runs make it difficult to relate the faster–than preorder of [5] to ours.

7 Conclusions and Future Work

We proposed a general amortised faster–than preorder for unifying bisimulation–based process theories [10, 11, 14] that relate asynchronous processes with respect to speed. Our amortised preorder ensures that a faster process must execute each action no later than the related slower process does, while both processes must be functionally equivalent in the sense of strong bisimulation [12].

Since the amortised faster–than preorder is normally not closed under all system contexts, we characterised the largest precongruences contained in it for a range of settings. The chosen range is spanned by a two–dimensional space, with one axis indicating whether only must–clock prefixes, only can–clock prefixes, or both are permitted, and the other axis determining whether only lazy actions, only urgent actions, or both kinds of actions are available. In this space, the settings of Moller/Tofts [14], which is concerned with lower time bounds, and of Lüttgen/Vogler [10], which is concerned with upper time bounds, can be recognised as "must/lazy" and "can/urgent" combinations, respectively. Since all reference preorders chosen in [10, 11] are less abstract than the amortised faster–than preorder, the results of this paper strengthen the ones obtained for both the Moller/Tofts and the Lüttgen/Vogler approach. The following table summarises our findings for each combination of clock prefix and action type, i.e., each entry identifies the behavioural relation that characterises the largest precongruence contained in the amortised faster–than preorder.

	Lazy	*Urgent*	*Lazy+Urgent*
Must	MT–preorder	Timed bisimulation	Urgent timed bisimulation
Can	Bisimulation	LV–preorder	LV–preorder
Must+Can	MT–preorder	Urgent timed bisimulation	Urgent timed bisimulation

The table shows that the amortised faster–than relation degrades to timed bisimulation as soon as must–clock prefixes and urgent actions come together. In this case, which includes the established process algebra TPL [7], one may express time intervals by equipping actions with both lower and upper time bounds. Moreover, when extending the Moller/Tofts approach by can–clock prefixing or the Lüttgen/Vogler approach by lazy actions, the MT–preorder and the LV–preorder, respectively, remain fully–abstract.

Future work shall investigate decision procedures for the MT– and LV–preorders, in order for them to be implemented in automated verification tools.

Acknowledgements. We would like to thank the anonymous referees for their valuable comments and suggestions.

References

[1] L. Aceto and D. Murphy. Timing and causality in process algebra. *Acta Inform.*, 33(4):317–350, 1996.
[2] S. Arun-Kumar and M.C.B. Hennessy. An efficiency preorder for processes. *Acta Inform.*, 29(8):737–760, 1992.
[3] S. Arun-Kumar and V. Natarajan. Conformance: A precongruence close to bisimilarity. In *STRICT '95*, Workshops in Comp., pp. 55–68. Springer-Verlag, 1995.
[4] F. Corradini, M. Di Berardini, and W. Vogler. PAFAS at work: Comparing the worst-case efficiency of three buffer implementations. In *APAQS 2001*, pp. 231–240. IEEE Computer Society Press, 2001.
[5] F. Corradini, R. Gorrieri, and M. Roccetti. Performance preorder and competitive equivalence. *Acta Inform.*, 34(11):805–835, 1997.
[6] R. Gorrieri, M. Roccetti, and E. Stancampiano. A theory of processes with durational actions. *TCS*, 140(1):73–94, 1995.
[7] M.C.B. Hennessy and T. Regan. A process algebra for timed systems. *Inform. and Comp.*, 117(2):221–239, 1995.
[8] L. Jenner and W. Vogler. Fast asynchronous systems in dense time. *TCS*, 254(1-2):379–422, 2001.
[9] G. Lüttgen and W. Vogler. Bisimulation on speed: A unified approach. Techn. Rep. 2004-15, Universität Augsburg, Germany, 2004.
[10] G. Lüttgen and W. Vogler. Bisimulation on speed: Worst–case efficiency. *Inform. and Comp.*, 191(2):105–144, 2004.
[11] G. Lüttgen and W. Vogler. Bisimulation on speed: Lower time bounds. *RAIRO Theoretical Informatics and Applications*, 2005. To appear.
[12] R. Milner. *Communication and Concurrency*. Prentice Hall, 1989.
[13] F. Moller and C. Tofts. A temporal calculus of communicating systems. In *CONCUR '90*, vol. 458 of *LNCS*, pp. 401–415. Springer-Verlag, 1990.
[14] F. Moller and C. Tofts. Relating processes with respect to speed. In *CONCUR '91*, vol. 527 of *LNCS*, pp. 424–438. Springer-Verlag, 1991.
[15] V. Natarajan and R. Cleaveland. An algebraic theory of process efficiency. In *LICS '96*, pp. 63–72. IEEE Computer Society Press, 1996.
[16] W. Vogler. Efficiency of asynchronous systems, read arcs, and the MUTEX-problem. *TCS*, 275(1–2):589–631, 2002.
[17] W. Vogler. Faster asynchronous systems. *Inform. and Comp.*, 184(2):311–342, 2003.