The MAY-preorder for timed systems with asynchronous communication

Giovanni Bernardi and Maryline Zhang
Université Paris Cité, CNRS, IRIF, F-75013, Paris, France
and
Pedro Ribeiro
Department of Computer Science, University of York, York, YO10 5GH, UK

In operational semantics after having defined how programs compute, one usually defines suitable preorders (or equivalences) to reason on programs. When reasoning on concurrent programs, typical refinements are Hennessy and De Nicola MAY-preorder and MUST-preorder.

The theory of MUST-preorder recently underwent a constructive overhaul which, thanks to an axiomatic approach, showed that the *standard* results for synchronous communication hold also for asynchronous communication, and in fact the synchronous theory is a special case of the asynchronous one. These results, though, pertain to a semantic model that does not account for time. Since timeouts are crucial to program distributed systems, not accounting for time is a serious limitation, which calls for an axiomatic study of testing preorders for (models of) systems with both time and communication via some shared medium (*i.e.* asynchronous communication).

This paper is a first step in this axiomatic study. Technically, we enrich the framework of [9, 11] to reason on time, and we show that the MAY-preorder coincides with inclusion of weak timed traces. We also show that this is *not* the case in a stronger model where we assume a typical property called *maximal progress*. This problem is independent from asynchronous communication, and it suggests that the standard techniques of testing theory do not work in a number of timed settings. In turn, this calls for a deeper study of this theory.

1 Introduction

To reason on a programming language via operational semantics, the first step is to define how programs compute, and the second is to define a preorder (or equivalence) to reason on programs in a compositional manner [4]. Defining preorders on programs is necessary also to have a proof technique for (i) correct software updates, that is to show that version 2 of a software can replace version 1 without causing any regression; (ii) iterative development, which, given a specification, lets us arrive at a correct by construction implementation by refining the specification in a step-wise manner.

One of the earliest such preorders was proposed in Morris' PhD thesis [35] to reason on deterministic higher-order programs, *i.e.* the λ-calculus. To reason on first-order *non-deterministic* programs, as we do in concurrency theory, two standard preorders are the so called MUST-preorder and MAY-preorder by De Nicola and Hennessy [25, 19, 26]. Both are defined contextually, that is by letting $p \leq q$ whenever for every test r if the parallel composition $p \mid r$ enjoys some property, *i.e.* $\phi(p \mid r)$, then so does $q \mid r$, *i.e.* $\phi(q \mid r)$. The property ϕ is typically some "observable" like termination, an error, a barb, etc ....

The MUST-preorder is formulated in a "positive" manner: one writes $p \sqsubseteq_{\text{MUST}} q$ whenever for *every* test r, if p passes r in *every* computation, then also q passes r in *every* computation. This preorder preserves test satisfaction, *i.e.* something *good*, and thus, at least intuitively, it preserves a liveness property ("something good must happen").

Submitted to:
EXPRESS/SOS 2026

In contrast, following [32], we present the MAY-preorder in a "negative" manner: first, we write $p \,\text{MAYF}\, r$ (read p may fail the test r) whenever in *some* computation of $p \mid r$ the test fails, *i.e.* it reaches a *bad* state. The reader familiar with unit testing or the use of assertions may recognise that $p \,\text{MAYF}\, r$ means that there exists an execution of $p \mid r$ in which some assertion in the test r is not satisfied, thereby raising some exception/error (*i.e.* "something bad happens"). Secondly, we write $p \sqsubseteq_{\text{MAYF}} q$ whenever for every test r, $p \,\text{MAYF}\, r$ implies that $q \,\text{MAYF}\, r$. This preorder is useful to ensure software safety, *i.e.* that nothing bad happens. The workflow is as follows: to being with, fix a *specification* q, and then, given an implementation p, to ensure that it is correct *wrt* q, it suffices to prove that $p \sqsubseteq_{\text{MAYF}} q$. If this is the case, then by definition all tests passed (*i.e.* not failed) by the specification q are passed (*i.e.* not failed) also by the implementation p, and thus nothing bad can happen by replacing the specification (q) with the implementation (p) [22].

The effective use of testing preorders is hampered by the universal quantification on tests used in their definitions. The first problem we face working with testing theory (and contextual relations in general) is therefore *the definition* of an alternative preorder $\preccurlyeq_{alt}$ such that

(i) the definition gives a proof method that does away with the universal quantification on tests;
(ii) the preorder $\preccurlyeq_{alt}$ is *sound* wrt the contextual preorder, *i.e.* $\preccurlyeq_{alt} \subseteq \sqsubseteq_{\text{MAYF}}$,
(iii) and also complete, *i.e.* $\sqsubseteq_{\text{MAYF}} \subseteq \preccurlyeq_{alt}$.

An alternative terminology for (ii) and (iii) is that $\preccurlyeq_{alt}$ is *adequate* and *fully-adequate wrt* $\sqsubseteq_{\text{MAYF}}$.

The authors of [9] have proven that the three *standard* alternative preorders for $\sqsubseteq_{\text{MUST}}$ are fully-adequate also in presence of communication via a shared medium (*i.e.* asynchronous communication). Two of these standard preorders are based respectively on comparing acceptance sets [26], and must-sets [19], while the third is the failure-divergence refinement [31, 15, 42]. The result is in fact stronger: [11] shows that the proofs of full-adequacy in case of synchronous communication are merely a special case of the ones for the case of asynchronous communication. This suggests that in studying testing preorders the standard definitions should be used as much as possible, possibly amending the LTS of programs and in this paper we follow this approach. For instance, for asynchrony it suffices to use Honda and Tokoro forwarding [29].

One key step in our technical development is the use of axioms over LTSs à la Selinger [47]: instead of fixing any particular calculus we model the behaviours of *both* programs and the shared communication medium used to exchange messages via two sets of axioms on LTSs [9, 11], namely FDBAX and FWDAX, that we present in Figure 2 and Figure 3. As we will see, the axioms FDBAX described the behaviours of programs, and are modelled *e.g.* by the early style LTSs of calculi like asynchronous CCS with value passing (VACCS); the axioms FWDAX instead describe the behaviours of programs together with a communication medium, and are modelled *e.g.* by the LTS of VACCS enhanced with forwarding à la Honda and Tokoro [29].

In this paper we combine the axioms FDBAX and FWDAX of [9, 11] with the axioms TIMEAX given in Figure 1. They are standard axioms to model the passage of time, discussed for instance in [1, Section 9.2], and ensure that in the formal model the passage of time is deterministic, can always be split in smaller delays, and that every process can let 0 time units pass. We thus work in a setting where

(a) processes are states in timed labelled transition systems, and
(b) communication between programs and tests takes place by storing messages in a shared communication medium, thereby being asynchronous,

and we show that the standard alternative characterisation of the MAY-preorder works (Theorem 1):

$$\forall p, q \models \text{FWDAX} \wedge \text{TIMEAX}.\ p \sqsubseteq_{\text{MAYF}} q \text{ iff } p \preccurlyeq_{tr} q \tag{1}$$

where $\preccurlyeq_{tr}$ denotes the inclusion of timed weak traces. Since the impact of the combination of the two families of axioms (respectively for asynchronous communication and for time passage) on standard results and proof techniques is a priori not clear, Equation (1) sheds light on the matter. Note that using FWDAX in Equation (1) seems necessary, because the standard technique (*i.e.* trace inclusion) does not work for the models of FDBAX, for instance in asynchronous CCS we have $a.\overline{a} \sqsubseteq_{\text{MAYF}} 0$, and $a.\overline{a} \stackrel{a}{\Longrightarrow} \overline{a}$ while $0 \stackrel{a}{\not\Longrightarrow}$. This is coherent with the results of [9, 11].

A priori Equation (1) suggests that axioms for asynchronous communication and axioms for time can be combined without breaking standard techniques. In reality, though, the situation is more subtle. Consider the following MAXIMAL-PROGRESS axiom:

$$\forall p.p \xrightarrow{\tau} \text{ implies } \forall d \in \mathbb{R}_{>0}.\neg(p \xrightarrow{d}). \tag{2}$$

where as usual $p \xrightarrow{\tau}$ represents a step of computation, and $p \xrightarrow{d}$ means that p can let d units of time pass. This axiom establishes a connection between passage of time and computation: as long as a process p can perform some calculation, it cannot let time pass. As we will see, adding MAXIMAL-PROGRESS to the axioms requires changing the standard semantics of parallel composition to ensure that the compositions $p \mid r$ satisfy the axiom. It turns out that when using this alternative semantics of parallel composition, neither a key technical property (Lemma 5), nor the standard characterisation of the MAY-preorder hold. In particular, we show that

$$\exists p,q \models \text{TIMEAX} \wedge \text{MAXIMAL-PROGRESS}.\ p \preccurlyeq_{tr} q \text{ and } \neg(p \sqsubseteq_{\text{MAYF}} q). \tag{3}$$

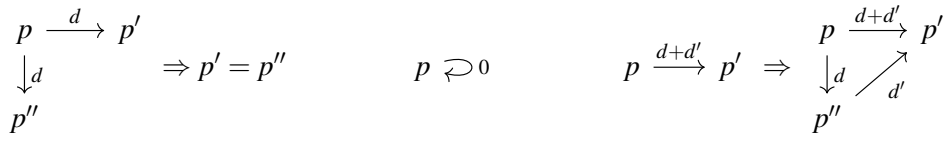
that is, trace inclusion is not sound *wrt* $\sqsubseteq_{\text{MAYF}}$. This is surprising, because adding axioms may reduce the expressivity of tests, thereby breaking the completeness result, not the soundness one (the empty set is trivially sound *wrt* $\sqsubseteq_{\text{MAYF}}$). Indeed, we will see that the root cause of the problem is the necessary change in the semantics of parallel composition. In passing, note that this lets us present a counterexample to [49, Theorem 1].

Contributions: Technically, the result of this paper is Equation (1). Owing to our axiomatic approach, to prove it in Table 1 we define a set of axioms sufficient for the completeness argument to go through. To the best of our knowledge both Equation (1) and our set of axioms is novel. Scientifically, though, we believe that the troubles caused by MAXIMAL-PROGRESS, and in particular Equation (3), are the main contribution of this submission. Example 5 explains why Equation (3) holds, and show thereby that that the study of testing theory in a timed setting warrants particular care, especially the study of the MUST-preorder, because its definition relies (implicitly) on an assumption of MAXIMAL-PROGRESS.

Paper structure: In Section 2 we present the axioms of the semantic model, we explain the most important ones, and we recall the semantics of parallel composition. In Section 3 we define and study the notion of weak timed traces that our result relies on, and we lay bare the relation between these traces and parallel composition (Lemmas 5 and 4). In Section 4 we define the MAY-preorder, and then state and prove our main result (Theorem 1). We present the axioms for the proof of completeness in Table 1. We devote Section 5 to the difficulties caused by the maximal progress axiom. In Section 6 we touch on related work, and we conclude this paper in Section 7.

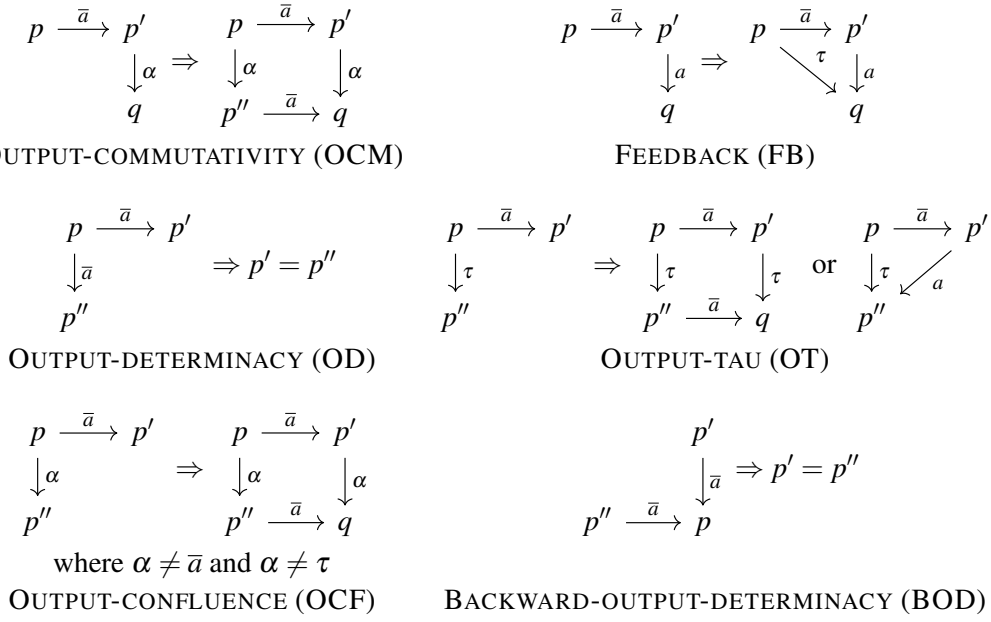
2 Axioms and parallel composition

We introduce the semantic model in which we work. We assume a countable set of names $\mathcal{N}$, ranged over by $a,b,c,\ldots$, and an analogous set of co-names, $\overline{\mathcal{N}}$, ranged over by $\overline{a},\overline{b},\overline{c},\ldots$. We further let



TIME-DETERMINACY ZERO-DELAY TIME-ADDITIVITY

Figure 1: Time axioms.



OUTPUT-COMMUTATIVITY (OCM) FEEDBACK (FB)

OUTPUT-DETERMINACY (OD) OUTPUT-TAU (OT)

OUTPUT-CONFLUENCE (OCF) BACKWARD-OUTPUT-DETERMINACY (BOD)

Figure 2: Axioms for asynchrony.

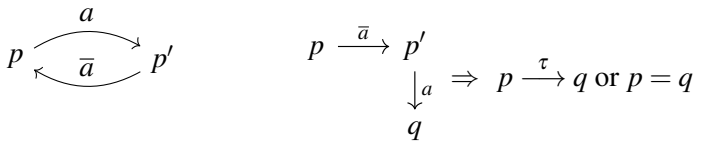
$\mathsf{Act} = \mathscr{N} \cup \overline{\mathscr{N}}$, $\mathsf{Act}_\tau = \mathsf{Act} \cup \{\tau\}$, and $L = \mathsf{Act}_\tau \cup \mathbb{R}_{\geq 0}$. We refer to L as the set of *labels*. We let μ range over Act, α range over L, d over $\mathbb{R}_{\geq 0}$, and λ over $(\mathsf{Act} \cup \mathbb{R}_{>0})$. We assume an involutive bijection $\bar{\cdot} : (\mathsf{Act} \cup \mathbb{R}_{>0}) \to (\mathsf{Act} \cup \mathbb{R}_{>0})$ where $\overline{d} = d$.

In order to define transitions systems we rely on the three sets of axioms that are depicted Figure 1, Figure 2, and Figure 3. For brevity the quantifications are omitted, assuming by convention that the quantifiers on the left-hand side of each axiom are universal, while on the right-hand side are existential. The only exception is axiom INPUT-BOOMERANG, where the quantification is $\forall p \forall a \exists p'$. Let TIMEAX denote the conjunction of the axioms in Figure 1. They are standard [1] so do not discuss them.

Definition 1 (TLTS)**.** *A timed labelled transitions system (TLTS) is a triple $\mathscr{L} = \langle A, L, \longrightarrow \rangle$, where A is a set of states, L is the set of labels defined above, and $\longrightarrow \subseteq A \times L \times A$ is a labelled transition relation that satisfies the axioms* TIMEAX *(*i.e. $\mathscr{L} \models$ TIMEAX*).*

When convenient, we use the subscript notation $\mathscr{L}_X$ to say that a given TLTS $\mathscr{L}$ has set of states X .

We now move to the axioms for asynchronous input/outputs that are given in Figure 2 and whose conjunction we denote FDBAX. They are Selinger axioms for output-buffered agents with feedback [47] plus BACKWARD-OUTPUT-DETERMINACY. Since they are amply discussed in [9, 8], here we comment just on OUTPUT-COMMUTATIVITY and OUTPUT-TAU. The OUTPUT-COMMUTATIVITY axiom ensures



INPUT-BOOMERANG (IB) FWD-FEEDBACK (FFB)

Figure 3: Axioms replacing FEEDBACK axiom for output-buffered agents with forwarding.

that processes can perform only outputs that are always "in parallel" with the actual control part of the process. This is akin to stating that any output message must be contained in a memory external to the actual code. Note that this is typical of standard imperative languages where the control (*i.e.* the program) modifies a state according to some reduction semantics [28, 14, 37, 41, 4]. The OUTPUT-TAU states that whenever a process performs a τ-transition it can be because either

(1) it performs some internal computations (the unfolding of a loop, and the choice of branch in a conditional, ...); or
(2) it performs a side-effect, *i.e.* thanks to an input it consumes a message that is in the memory.

Note that in Figure 2 and Figure 3 the labels pertain just to (co-)names. If a set of values is defined, it is obvious how to generalise the axioms to labels carrying values, *i.e.* $a?u, a!u$ or $(a,u), \overline{(a,u)}$.

The axioms in Figure 2 are powerful enough to capture the semantics of calculi like asynchronous CCS with value passing (VACCS). First of all, since in asynchronous calculi outputs have no continuation and can be written only in parallel, we have the following fact.

Lemma 1. *For every $p \in$ VACCS there exists a $p' \in$ VACCS and a possibly empty multiset of outputs $\{|a_1!u_1, \ldots, a_n!u_n|\}$ such that*

(a) $p \equiv p' \mid a_1!u_1 \mid \ldots \mid a_n!u_n$, *and*
(b) $p' \xrightarrow{\alpha}$ *implies* α *is not an output.*

For a proof see [44, Lemma 5.3.1] or [8, Lemma 2.17]. A consequence of Lemma 1 is the property we announced.

Lemma 2. *The early style LTS of VACCS modulo structural equivalence is a model of the axioms* FDBAX.

This lemma is discussed in [11], its Rocq proof is available on-line[1], and the argument is essentially the one for asynchronous CCS given in [8].

The reason why the early style LTS of VACCS enjoys OUTPUT-COMMUTATIVITY is Lemma 1, which suggests that the axioms FDBAX capture the behaviour of a program (the control) together with some sort of "container" for the data. In the statement of the lemma this "container" is the multiset of outputs. Notice that the "container" does not really model a memory, because one crucial feature is missing: a memory (*i.e.* communication medium) is *input-enabled*. To start with, in real life anybody can put a letter in a mailbox, *i.e.* physical mailboxes are input-enabled. The reader familiar with IO-automata [33] will not be surprised by requiring the communication medium to be input-enabled, neither will be the reader familiar with concurrent objects [30] or with the forwarding proposed by Honda and Tokoro [29]. Forwarding is a technique used by the π-calculus community to reason on (variants of) the asynchronous CCS and π-calculus [29, 27, 9, 11], and it is exactly what adds the input-enabledness to the syntactic

[1] Available at https://glmaths.github.io/testing-theory/toc.html.

terms of calculi, thereby producing an LTS that satisfies INPUT-BOOMERANG. The details can be found in [9, Lemma 2].

We axiomatise the behaviour of a program together with an input-enabled memory by letting FWDAX be the conjunction of the axioms in Figure 3 together with the ones in Figure 2 *except* FEEDBACK. The axiom INPUT-BOOMERANG guarantees that in every execution a process can always perform any input (*i.e.* it is input-enabled), and if it does then it changes state (the memory is mutable). The state that it reaches is ready to output the datum stored in the memory by performing the input, and if it does, the memory reverts the state preceding the input. The INPUT-BOOMERANG axiom holds for the LTSs of multisets (add/remove), stacks (push/pop), and the ordinary state (set/get). Multisets are of interest to reason on distributed systems, because they model communication via `UDP`, which does not retain the message order. Consider the following example.

Example 1. *We present the LTS of an unordered mutable memory. Let* Val *be a set of values, and suppose we want to reason on the content of a shared memory in which* $\mathscr{N}$ *are the memory locations (in our particular case, channels), and each location may contain a finite unordered amount of values. We model this memory via elements of the function space*

$$MO = \mathscr{N} \to (\mathsf{Val} \to \mathbb{N})$$

We let the symbols $M, N, \ldots$ *range over MO. Note that the elements of MO have the same domain of the usual state for imperative languages,* i.e. *memory locations, with the difference that, to reason on* `UDP`*, as codomain we have to use multisets of values instead of a single value. In our setting the empty memory is the constant* $\lambda x \lambda y.0$*.*

A handier notation to represent memory states in examples is the standard multiset notation, i.e. $\{|\ldots|\}$*. For instance,*

$$\varnothing, \{|(a,1)|\}, \{|(a,9),(a,9)|\}, \{|(a,3),(b,5),(a,7),(b,5)|\} \in MO.$$

denote respectively the memory that is empty, that contains 1 *in channel a, that contains two occurrences of the value* 9 *in channel a, ... We also denote with* $\uplus$ *the multiset union.*

The way in which a multiset interacts with the environment is formalised by the following transition rules, for every $a \in \mathscr{N}$ *and* $u \in \mathsf{Val}$*:*

(1) $M \xrightarrow{(a,u)}_{\mathsf{m}} M \uplus \{|(a,u)|\}$*; and*

(2) $M \uplus \{|(a,u)|\} \xrightarrow{\overline{(a,u)}}_{\mathsf{m}} M$*.*

*Rule (1) states that a memory can perform any input (*i.e. *it is input-enabled), thereby changing its state by adding to the appropriate location the value read. Rule (2) states the dual: a memory can output any datum that it contains, thereby removing it. The LTS* $\langle MO, \mathsf{Act}, \longrightarrow_{\mathsf{m}} \rangle$ *enjoys* INPUT-BOOMERANG*. Note that this LTS does not contain* τ *transitions. This should be obvious: a communication medium (*i.e. *the memory) performs no computations, because it is merely passive and manipulated by programs. The LTS that we just defined is depicted in [20, Fig. 3.2] for a model of value passing where values are in* $\{0,1\}$ *and where there is only one channel that is thus omitted from the labels.* □

As for forwarding, we have the following fact [29].

Lemma 3. *The early style LTS of* `VACCS` *modulo structural equivalence, once enhanced with forwarding, is a model of the axioms* FWDAX*.*

Proof. The (mechanised) proof of this fact is discussed in [11]. □

$$\text{[PROC]}\ \frac{p \xrightarrow{\alpha} p'}{p \mid r \xrightarrow{\alpha} p' \mid r} \qquad \text{[TEST]}\ \frac{r \xrightarrow{\alpha} r'}{p \mid r \xrightarrow{\alpha} p \mid r'} \qquad \text{[COM]}\ \frac{p \xrightarrow{\mu} p' \quad r \xrightarrow{\overline{\mu}} r'}{p \mid r \xrightarrow{\tau} p' \mid r'} \qquad \text{[DELAY]}\ \frac{p \xrightarrow{d} p' \quad r \xrightarrow{d} r'}{p \mid r \xrightarrow{d} p' \mid r'}$$

Figure 4: The TLTS of server-client systems.

The axioms TIMEAX pertain to the transitions labelled by delays, while FDBAX and FWDAX give conditions on inputs and outputs, and are thereby orthogonal to TIMEAX.

We now combine them by defining the two sets of TLTS that we study in the rest of the paper:

$$\begin{array}{lcl} \text{TFDB} & = & \{\mathscr{L} \mid \mathscr{L} \models \text{FDBAX} \wedge \text{TIMEAX}\}, \\ \text{TFWD} & = & \{\mathscr{L} \mid \mathscr{L} \models \text{FWDAX} \wedge \text{TIMEAX}\}. \end{array} \tag{4}$$

Now that we have established the TLTS that we are concerned with, we introduce the parallel composition over them via the rules in Figure 4. They are standard, and rule [DELAY] ensures that physical time passes uniformly for all the processes that run concurrently. Even though the semantics is standard, the asymmetric notation $p \mid r$ is not. We use it throughout this paper to highlight two asymmetries: (i) the process p belongs to FWDAX, *i.e.* its behaviour models the communication medium, while r belongs to FDBAX; (ii) the process p is the process tested by the test r. To formally define the MAY-preorder we focus on the transitions labelled with τ or with a delay: given a TLTS $\langle p \mid r, \{\tau\} \cup \mathbb{R}_{\geq 0}, \longrightarrow \rangle$, we say that a *computation* of $p \mid r$ is a possibly infinite sequence of transitions that starts from $p \mid r$, *i.e.*

$$p_0 \mid r_0 \xrightarrow{\eta_0} p_1 \mid r_1 \xrightarrow{\eta_1} p_2 \mid r_2 \xrightarrow{\eta_2} \ldots$$

where $p \mid r = p_0 \mid r_0$ and $\eta_i \in \{\tau\} \cup \mathbb{R}_{\geq 0}$.

Example 2 (Communication via shared medium)**.** *We now outline how a model of the axioms* FWDAX *can be used to model the interaction between programs that communicate via a shared communication channel. Let*

$$\begin{array}{lcl} p & \equiv & \texttt{let x = rcv(a) in ()} \\ r & \equiv & \texttt{send(a,3);()} \end{array}$$

where a *is some port, for instance* 8080*. To express that p can input any value over* a *we give it the LTS* $p \xrightarrow{(\mathtt{a},\mathtt{v})}_{\mathtt{c}} \texttt{()}$ *for every v, and to model that r performs the output denoted by its syntax we give it the LTS* $r \xrightarrow{\overline{(\mathtt{a},3)}} \texttt{()}$*. In the LTS of p the subscript* c *is a reminder that the transitions are of the control. As typical when using reduction semantics, we want to define the computations of p together with a mutable state,* i.e. *the communication medium. This is done via the parallel composition of LTS* à la CCS*, which lets us compose the LTS of p with the one* $\varnothing$ *defined in Example 1. Consider the term* $p \mid \varnothing$*, one possible execution is as follows:*

$$(p \mid \varnothing) \xrightarrow{(\mathtt{a},\mathtt{v})}_{\mathtt{m}} (p \mid \{|(\mathtt{a},\mathtt{v})|\}) \xrightarrow{\tau} (\texttt{()} \mid \varnothing)$$

The left-most transition is the input whereby the medium stores a datum, and the τ *transition is the result of the interaction between p, which is ready to input on* a *and the medium that is ready to output on the same port. Note that the transition by the memory is cunningly tagged with* m*. To represent the interactions of an external observer, for instance a test, with p through the medium it suffices to hide the actions of p. To this end we use a version of the restriction operation* ν *that operates only on the*

transitions tagged with $\mathtt{c}$*, and consider* $q = (\nu \mathtt{a})(p \mid \varnothing)$*. Thanks to the restriction, the only interactions that q offers to the environment (*i.e. *the tests) are the ones offered by the medium, which thus becomes the interface of q. First, the LTS of this system belongs to the class* FWDAX*. Second, thanks to the input transitions of the medium (which are visible), we obtain the following computation:*

$$q \mid r \xrightarrow{\tau} (\nu \mathtt{a})(p \mid \{\!|(\mathtt{a},3)|\!\}) \mid () \xrightarrow{\tau} (\nu \mathtt{a})(() \mid \varnothing) \mid ()$$

These two transitions show how the value passes from the program r to the program p, which is inside to the process q, via the medium. □

3 Weak traces and (un)zipping

The statement of our result (Theorem 1) depends on the notion of weak timed trace, which we introduce and discuss in this section. Let $\mathscr{W} = \{s \mid s \in (\mathsf{Act} \cup \mathbb{R}_{>0})^\star$ and s has no two consecutive delays$\}$. By abuse of notation, we write $0.s$ to mean s. Given a TLTS $\langle A, L, \longrightarrow \rangle$, we define the weak transition relation $\Longrightarrow \subseteq A \times \mathscr{W} \times A$ as the least one that satisfies the following rules:

[WT-REFL] $p \overset{\varepsilon}{\Longrightarrow} p$,
[WT-TAU] $p \overset{s}{\Longrightarrow} q$ if $p \xrightarrow{\tau} p'$ and $p' \overset{s}{\Longrightarrow} q$ for some $p' \in A$,
[WT-MU] $p \overset{\mu s}{\Longrightarrow} q$ if $p \xrightarrow{\mu} p'$ and $p' \overset{s}{\Longrightarrow} q$ for some $p' \in A$,
[WT-TIME] $p \overset{d.s}{\Longrightarrow} q$ if $p \overset{d-d'}{\longrightarrow} p'$, $p' \overset{d'.s}{\Longrightarrow} q$, and $d \neq d'$ for some $d' \in \mathbb{R}_{\geq 0}$ and $p' \in A$.

In [WT-TIME] rule, note that $d > 0$ as $d - d' > 0$ by $d \neq d'$. As per usual, we write $p \overset{s}{\Longrightarrow}$ to mean $\exists p'.\ p \overset{s}{\Longrightarrow} p'$. We now digress on a little technicality that will be useful to arrive at proofs by induction.

Example 3. *Let* $\mathscr{L}_A$ *be a TLTS such that* $p \xrightarrow{1} q$ *for some* $p, q \in A$*. By* TIME-ADDITIVITY*, there exists* $p' \in A$ *such that* $p \xrightarrow{0.1} p' \xrightarrow{0.9} q$*. Then we prove that* $p \overset{1}{\Longrightarrow} q$ *via the following derivation trees:*

$$\dfrac{p \xrightarrow{1} q \qquad \dfrac{}{q \overset{\varepsilon}{\Longrightarrow} q}\,[\text{WT-REFL}]}{p \overset{1}{\Longrightarrow} q}\,[\text{WT-TIME}] \qquad\qquad \dfrac{p \xrightarrow{0.1} p' \qquad \dfrac{p' \xrightarrow{0.9} q \qquad \dfrac{}{q \overset{\varepsilon}{\Longrightarrow} q}\,[\text{WT-REFL}]}{p' \overset{0.9}{\Longrightarrow} q}\,[\text{WT-TIME}]}{p \overset{1}{\Longrightarrow} q}\,[\text{WT-TIME}]$$

■

Example 3 shows that, owing to time additivity, $p \overset{s}{\Longrightarrow} p'$ can be proven via several derivation trees, and this hampers proofs by direct rule induction. We can still reason inductively as follows: we let $|p \overset{s}{\Longrightarrow} p'|$ denote the minimal height of all the derivation trees of $p \overset{s}{\Longrightarrow} p'$. This minimum is well-defined because $p \overset{s}{\Longrightarrow} p'$ means that at least one derivation exists, the height of every derivation tree is in $\mathbb{N}$, and $(\mathbb{N}, <)$ is well-founded.

Our definition of weak traces is not standard. In spite of this definitional difference, our predicate $\overset{s}{\Longrightarrow}$ equals the one introduced in [1, Definition 11.8]. This fact is proved in Appendix A.

We now state and prove two lemmas that establish the logical link between the semantics of parallel composition, the notion of computation, and weak traces. They are the unzipping of computations into weak traces, and the converse, *i.e.* zipping. These lemmas are crucial to show Theorem 1. Albeit being standard, the presence of delays in transitions makes the proofs somewhat delicate, and in Section 5 we will see that since the assumption of MAXIMAL-PROGRESS forces us to change the semantics of parallel composition, adding MAXIMAL-PROGRESS makes Lemma 5 false.

Lemma 4 (Unzipping). *For every TLTS $\mathscr{L}_A$, $\mathscr{L}_B$, $p,p' \in A$, $r,r' \in B$, $d \in \mathbb{R}_{\geq 0}$,*

$$p \mid r \stackrel{d}{\Longrightarrow} p' \mid r' \text{ implies there exists } s \in \mathscr{W} \text{ such that } p \stackrel{s}{\Longrightarrow} p' \text{ and } r \stackrel{\overline{s}}{\Longrightarrow} r'.$$

Proof. We proceed by induction on the derivation of $p \mid r \stackrel{d}{\Longrightarrow} p' \mid r'$. The inductive case [WT-MU] is trivial. In the base case [WT-REFL], $p' \mid r' = p \mid r$, we choose $s = \varepsilon$, and we conclude by [WT-REFL].

In the inductive case [WT-TAU] $p \mid r \stackrel{\tau}{\longrightarrow} \hat{p} \mid \hat{r} \stackrel{d}{\Longrightarrow} p' \mid r'$ for some $\hat{p} \in A$ and $\hat{r} \in B$, the inductive hypothesis states that $\hat{p} \stackrel{s'}{\Longrightarrow} p'$ and $\hat{r} \stackrel{\overline{s'}}{\Longrightarrow} r'$ for some $s' \in \mathscr{W}$. Then we continue by case analysis on the rule used to derive the transition $p \mid r \stackrel{\tau}{\longrightarrow} \hat{p} \mid \hat{r}$. In the case [PROC], we have that $p \stackrel{\tau}{\longrightarrow} \hat{p}$ and $\hat{r} = r$. Then it follows from [WT-TAU] that $p \stackrel{s'}{\Longrightarrow} p'$, thus we let $s = s'$. In the case [TEST], we have that $r \stackrel{\tau}{\longrightarrow} \hat{r}$ and $\hat{p} = p$. Then by [WT-TAU], we know that $r \stackrel{\overline{s'}}{\Longrightarrow} r'$, thus we take $s = s'$. In the case [COM], we have that $p \stackrel{\mu}{\longrightarrow} \hat{p}$ and $r \stackrel{\overline{\mu}}{\longrightarrow} \hat{r}$ for some $\mu \in \mathsf{Act}$. Then by [WT-MU], we obtain that $p \stackrel{\mu.s'}{\Longrightarrow} p'$ and $r \stackrel{\overline{\mu.s'}}{\Longrightarrow} r'$, thus we let $s = \mu.s'$.

In the inductive case [WT-TIME], we have that $d \in \mathbb{R}_{>0}$ and $p \mid r \stackrel{d-d'}{\longrightarrow} \hat{p} \mid \hat{r} \stackrel{d'}{\Longrightarrow} p' \mid r'$ for some $d' \in \mathbb{R}_{\geq 0}$, $\hat{p} \in A$ and $\hat{r} \in B$. The inductive hypothesis gives that $\hat{p} \stackrel{s'}{\Longrightarrow} p'$ and $\hat{r} \stackrel{\overline{s'}}{\Longrightarrow} r'$ for some $s' \in \mathscr{W}$. By case analysis, we deduce that $p \stackrel{d-d'}{\longrightarrow} \hat{p}$ and $r \stackrel{d-d'}{\longrightarrow} \hat{r}$. Then, the results are given by [WT-TIME]. □

Lemma 5 (Zipping). *For every TLTS $\mathscr{L}_A$, $\mathscr{L}_B$, $p,p' \in A$, $r,r' \in B$, $s \in \mathscr{W}$,*

$$p \stackrel{s}{\Longrightarrow} p' \text{ and } r \stackrel{\overline{s}}{\Longrightarrow} r' \text{ imply there exists } d \in \mathbb{R}_{\geq 0} \text{ such that } p \mid r \stackrel{d}{\Longrightarrow} p' \mid r'.$$

Proof. We proceed by complete induction on $|p \stackrel{s}{\Longrightarrow} p'| + |r \stackrel{\overline{s}}{\Longrightarrow} r'|$. In the base case $|p \stackrel{s}{\Longrightarrow} p'| + |r \stackrel{\overline{s}}{\Longrightarrow} r'| = 0$, we have that $p' = p$ and $r' = r$, thus the conclusion follows by [WT-REFL].

In the inductive case $|p \stackrel{s}{\Longrightarrow} p'| + |r \stackrel{\overline{s}}{\Longrightarrow} r'| = n+1$ for some $n \in \mathbb{N}$, the inductive hypothesis states that for every $p_1, p_2 \in A$, $r_1, r_2 \in B$, $s' \in \mathscr{W}$,

$$|p_1 \stackrel{s'}{\Longrightarrow} p_2| + |r_1 \stackrel{\overline{s'}}{\Longrightarrow} r_2| \leq n \text{ implies there exists } d' \in \mathbb{R}_{\geq 0} \text{ such that } p_1 \mid r_1 \stackrel{d'}{\Longrightarrow} p_2 \mid r_2. \qquad (IH)$$

We continue by case analysis on the rule used to derive the reduction $p \stackrel{s}{\Longrightarrow} p'$.

In the case [WT-REFL], we have that $p' = p$, $s = \varepsilon$ and $|p \stackrel{\varepsilon}{\Longrightarrow} p'| = 0$. Then we proceed by case analysis on the rule used to derive the reduction $r \stackrel{\varepsilon}{\Longrightarrow} r'$. The cases [WT-REFL], [WT-MU], and [WT-TIME] are trivial. In the case [WT-TAU], we know that $r \stackrel{\tau}{\longrightarrow} \hat{r}$ and $|\hat{r} \stackrel{\varepsilon}{\Longrightarrow} r'| = n$ for some $\hat{r} \in B$. Then by (IH), we deduce that $p \mid \hat{r} \stackrel{d'}{\Longrightarrow} p' \mid r'$ for some $d' \in \mathbb{R}_{\geq 0}$, and the conclusion follows from [TEST].

In the case [WT-TAU], we have that $p \stackrel{\tau}{\longrightarrow} \hat{p}$, $|\hat{p} \stackrel{s}{\Longrightarrow} p'| = n - m$, $|r \stackrel{\overline{s}}{\Longrightarrow} r'| = m$ and $m \leq n$ for some $m \in \mathbb{N}$ and $\hat{p} \in A$. Therefore, $\hat{p} \mid r \stackrel{d'}{\Longrightarrow} p' \mid r'$ for some $d' \in \mathbb{R}_{\geq 0}$ follows by (IH), then the result is deduced from [PROC].

In the case [WT-MU], we have that $s = \mu.s'$, $p \stackrel{\mu}{\longrightarrow} \hat{p}$, $|\hat{p} \stackrel{s'}{\Longrightarrow} p'| = n - m$, $|r \stackrel{\overline{\mu.s'}}{\Longrightarrow} r'| = m$ and $m \leq n$ for some $m \in \mathbb{N}$, $\mu \in \mathsf{Act}$, $s' \in \mathscr{W}$ and $\hat{p} \in A$. Then we proceed by case analysis on the rule used to derive the reduction $r \stackrel{\overline{\mu.s'}}{\Longrightarrow} r'$. The cases [WT-REFL] and [WT-TIME] are trivial. In the case [WT-TAU], we know that $r \stackrel{\tau}{\longrightarrow} \hat{r}$ and $|\hat{r} \stackrel{\overline{\mu.s'}}{\Longrightarrow} r'| = m - 1$ for some $\hat{r} \in B$. Then (IH) implies that $p \mid \hat{r} \stackrel{d'}{\Longrightarrow} p' \mid r'$ for some $d' \in \mathbb{R}_{\geq 0}$, and the conclusion follows from [TEST]. In the case [WT-MU], we know that $r \stackrel{\mu}{\longrightarrow} \hat{r}$ and $|\hat{r} \stackrel{\overline{s'}}{\Longrightarrow} r'| = m - 1$

for some $\hat{r} \in B$. Then from (IH), we obtain that $\hat{p} \mid \hat{r} \stackrel{d'}{\Longrightarrow} p' \mid r'$ for some $d' \in \mathbb{R}_{\geq 0}$, and we conclude by [COM].

In the case [WT-TIME], we have that $s = d.s'$, $p \stackrel{d-d'}{\longrightarrow} \hat{p}$, $|\hat{p} \stackrel{d'.s'}{\Longrightarrow} p'| = n-m$, $|r \stackrel{\overline{d.s'}}{\Longrightarrow} r'| = m$ and $m \leq n$ for some $m \in \mathbb{N}$, $d \in \mathbb{R}_{>0}$, $d' \in \mathbb{R}_{\geq 0}$, $s' \in \mathscr{W}$ and $\hat{p} \in A$. Then we proceed by case analysis on the rule used to derive the reduction $r \stackrel{\overline{d.s'}}{\Longrightarrow} r'$. The cases [WT-REFL] and [WT-MU] are trivial. In the case [WT-TAU], we know that $r \stackrel{\tau}{\longrightarrow} \hat{r}$ and $|\hat{r} \stackrel{\overline{d.s'}}{\Longrightarrow} r'| = m-1$ for some $\hat{r} \in B$. Then it follows from (IH) that $p \mid \hat{r} \stackrel{d_1}{\Longrightarrow} p' \mid r'$ for some $d_1 \in \mathbb{R}_{\geq 0}$ and the result is given by [TEST]. In the case [WT-TIME], we know that $r \stackrel{d-\hat{d}}{\longrightarrow} \hat{r}$ and $|\hat{r} \stackrel{\overline{\hat{d}.s'}}{\Longrightarrow} r'| = m-1$ for some $\hat{d} \in \mathbb{R}_{>0}$ and $\hat{r} \in B$. If $d' = \hat{d}$ then $\hat{p} \mid \hat{r} \stackrel{d_1}{\Longrightarrow} p' \mid r'$ for some $d_1 \in \mathbb{R}_{\geq 0}$ by (IH), and we conclude by [DELAY]. If $d' < \hat{d}$ then we obtain that $p \stackrel{d-\hat{d}}{\longrightarrow} p_1 \stackrel{\hat{d}-d'}{\longrightarrow} \hat{p}$ for some $p_1 \in A$ by the TIME-ADDITIVITY axiom, then we have that $|p_1 \stackrel{\hat{d}.s'}{\Longrightarrow} p'| = n-m+1$ by [WT-TIME]. Thus, (IH) implies that $p_1 \mid \hat{r} \stackrel{d_1}{\Longrightarrow} p' \mid r'$ for some $d_1 \in \mathbb{R}_{\geq 0}$, and the conclusion follows from [DELAY]. If $\hat{d} < d'$ then we obtain that $r \stackrel{d-d'}{\longrightarrow} r_1 \stackrel{d'-\hat{d}}{\longrightarrow} \hat{r}$ for some $r_1 \in B$ by the TIME-ADDITIVITY axiom, then we have that $|r_1 \stackrel{\overline{d'.s'}}{\Longrightarrow} r'| = m$ by [WT-TIME]. Thus, (IH) implies that $\hat{p} \mid r_1 \stackrel{d_1}{\Longrightarrow} p' \mid r'$ for some $d_1 \in \mathbb{R}_{\geq 0}$, and the conclusion follows from [DELAY]. □

We conclude this section by stating in two lemmas (proved in Appendix B) that the initial and final states of weak timed traces are invariant under four manipulations on the traces themselves. The first lemma shows the impact of FEEDBACK: a pair of labels comprising an output followed (weakly) by the respective input can be omitted. The second lemma ensures that in a trace, we can: postpone any label μ, make appear a name and its co-name, and merge delays, without changing the state reached by the trace. More formally, suppose that $\mathscr{L}_A$ satisfies OUTPUT-COMMUTATIVITY and FWD-FEEDBACK, and that $p, p', q \in A$. Then we have the following.

Lemma 6 (Feedback)**.** *For every $a \in \mathscr{N}$, $s \in \mathscr{W}$, $p \stackrel{\overline{a}}{\longrightarrow} p' \stackrel{a.s}{\Longrightarrow} q$ imply $p \stackrel{s}{\Longrightarrow} q$.*

Lemma 7 (Trace morphisms)**.** *If $\mathscr{L}_A \models$ IB, then for every $s_1 \in \mathscr{N}^\star$, and $s_3 \in \mathscr{W}$, we have*

1. *for every $\mu \in \mathsf{Act}$, $p \stackrel{\mu}{\longrightarrow} p' \stackrel{s_1.s_3}{\Longrightarrow} q$ imply $p \stackrel{s_1.\mu.s_3}{\Longrightarrow} q$.*
2. *for every $a \in \mathscr{N}$, $s_2 \in \mathscr{N}^\star$, $p \stackrel{s_1.s_2.s_3}{\Longrightarrow} p'$ implies $p \stackrel{s_1.a.s_2.\overline{a}.s_3}{\Longrightarrow} p'$.*
3. *for every $d, d' \in \mathbb{R}_{\geq 0}$, $p \stackrel{d}{\longrightarrow} p' \stackrel{s_1.d'.s_3}{\Longrightarrow} q$ imply $p \stackrel{s_1.(d+d').s_3}{\Longrightarrow} q$.*

We can now move on the main technical result of this paper.

4 The MAY-preorder coincides with trace inclusion

Our aim is to prove that whenever two processes p, q are in FWDAX and the tests are in FDBAX, we have $p \sqsubseteq_{\text{MAYF}} q$ if and only if the weak timed traces of p are also weak timed traces of q. First, we assume a predicate BAD over tests. Second, we say that a computation

$$p \mid r = p_0 \mid r_0 \stackrel{\eta_0}{\longrightarrow} p_1 \mid r_1 \stackrel{\eta_1}{\longrightarrow} p_2 \mid r_2 \stackrel{\eta_2}{\longrightarrow} \ldots$$

fails if there exists an $i \in \mathbb{N}$ such that $\text{BAD}(r_i)$, *i.e.* if something bad happens (to the test) in it.

In the following definition we slightly abuse the notation: for any TLTS $\mathscr{L}$ with set of states X, we write $r \in \mathscr{L}$ in place of $r \in X$.

$\forall s \in \mathscr{W}$, $\forall \mu \in \mathsf{Act}$, $\forall a \in \mathscr{N}$, $\forall d \in \mathbb{R}_{>0}$, $\forall 0 \leq d' \leq d$, $\forall \alpha \in (\mathsf{Act} \cup \mathbb{R}_{>0})$, $\forall r \in C$,

(1) $\textsc{bad}(tt(s))$ if and only if $s \in \mathscr{N}^\star$
(2) $tt(\varepsilon) \stackrel{\alpha}{\not\longrightarrow}$
(3) $tt(\mu.s) \xrightarrow{\overline{\mu}} tt(s)$
(4) $tt(\overline{a}.s) \xrightarrow{\alpha} r$ implies $\alpha = a$ and $r = tt(s)$
(5) $tt(d.s) \xrightarrow{d-d'} tt(d'.s)$
(6) $tt(d.s) \xrightarrow{\alpha} r$ implies $\alpha \leq d$ and $r = tt((d-\alpha).s)$

Table 1: Axioms for (the function that generates) tests to prove completeness.

Definition 2 (MAY-preorder)**.** *We write p $\textsc{mayf}_n$ r if there exists a computation of $p \mid r$ that is failed with $\textsc{bad}(r_n)$, and we write p MAYF r whenever p $\textsc{mayf}_n$ r for some n.*

We let $p \sqsubseteq^{\mathscr{L}}_{\text{MAYF}} q$ whenever p MAYF r implies q MAYF r for every test $r \in \mathscr{L}$.

Note that $\sqsubseteq^{\mathscr{L}}_{\text{MAYF}}$ denotes in fact a function of $\mathscr{L}$. This parametrisation of the contextual preorder under scrutiny *wrt* the TLTS of clients is analogous to [11, Definition 2], and it will be convenient to state the hypothesis sufficient to prove the completeness of the alternative characterisation, which we define next.

Definition 3 (Trace inclusion)**.** *We write $p \preccurlyeq_{tr} q$ whenever for every $s \in \mathscr{W}$, if $p \stackrel{s}{\Longrightarrow}$ then $q \stackrel{s}{\Longrightarrow}$.*

To characterise $\sqsubseteq_{\text{MAYF}}$ via $\preccurlyeq_{tr}$ we will use Lemma 4, Lemma 5 and one key property that pertains to suitable family of tests. The definition of this family is delicate. Since we do not work with any specific calculus, following [9, 11], instead of giving concrete tests, we present a set of axioms for a function that generates TLTS (*i.e.* tests), that are *sufficient* for our purposes.

Given a TLTS $\mathscr{L}_C$, we write $\mathscr{L}_C \models \textsc{AxCmpl}$ if the following conditions are true:

(1) $\mathscr{L}_C \models \text{OD} \wedge \text{OT} \wedge \text{OCF} \wedge \text{BOD}$, and
(2) there exist a predicate $\textsc{bad} : C \to \texttt{bool}$ and a function $tt : \mathscr{W} \to C$ that satisfy the axioms in Table 1.

The function tt entailed by AxCmpl generates the tests (*i.e.* TLTS) whose behavioural properties are described in Table 1, and that are necessary and sufficient for our purposes. As for their behaviours, thanks to the axioms (1), (3), and (5) in Table 1, a test $tt(s)$ can perform entirely the dual of s and thereby reach a BAD state (*i.e.* Lemma 8).

The axioms in Table 1 let us prove the technical Lemmas (8) - (11) and Proposition 1 about tests. We defer their proofs to Section 4.1. The next five facts apply to any TLTS $\mathscr{L}_C$ that satisfies AxCmpl, and for brevity we omit the universal quantification on $s \in \mathscr{W}$ from the statements.

Lemma 8. *There exists $r \in C$ such that $tt(s) \stackrel{\overline{s}}{\Longrightarrow} r$ and $\textsc{bad}(r)$.*

The next three facts pertain to traces that start with a finite prefix s_1 that contains only input actions.

Lemma 9. *For every $\mu \in \mathsf{Act}$, $r \in C$, if $tt(s) \xrightarrow{\mu} r$ then there exist two traces $s_1 \in \mathscr{N}^\star$ and $s_2 \in \mathscr{W}$ such that $s = s_1.\overline{\mu}.s_2$ and $r = tt(s_1.s_2)$.*

Lemma 10. *For every $r \in C$, if $tt(s) \xrightarrow{\tau} r$ then there exist three traces $s_1, s_2 \in \mathscr{N}^\star$, $s_3 \in \mathscr{W}$, and an action $a \in \mathscr{N}$ such that $s = s_1.a.s_2.\overline{a}.s_3$ and $r = tt(s_1.s_2.s_3)$.*

In the following lemma, the operator $\cdot$ denotes language concatenation. For instance, the somewhat unfriendly notation $\mathscr{W} \setminus (\mathbb{R}_{>0} \cdot \mathscr{W})$ merely denotes the set of all finite words over $\mathsf{Act} \cup \mathbb{R}_{>0}$ that (i) start with no delay, and (ii) have no two consecutive delays.

Lemma 11. *For every $d \in \mathbb{R}_{>0}$, $r \in C$, if $tt(s) \xrightarrow{d} r$ then there exist a trace $s_1 \in \mathscr{N}^\star$, a trace $s_2 \in \mathscr{W} \setminus (\mathbb{R}_{>0} \cdot \mathscr{W})$, and a delay $d' \in \mathbb{R}_{\geq 0}$ such that $s = s_1.(d+d').s_2$ and $r = tt(s_1.d'.s_2)$.*

We now prove the key property of the tests that satisfy the axioms in Table 1: the fact that a process p may fail a test $tt(s)$ is logically equivalent to p performing the weak timed trace s.

Proposition 1. *For every TLTS $\mathscr{L}_A \models \mathrm{OCM} \wedge \mathrm{FFB} \wedge \mathrm{IB}$, $p \in A$, $p \stackrel{s}{\Longrightarrow}$ iff p MAYF $tt(s)$.*

Proof. Suppose $p \stackrel{s}{\Longrightarrow}$. Since Lemma 8 implies that $tt(s) \stackrel{\bar{s}}{\Longrightarrow} r$ and BAD(r) for some $r \in C$, the conclusion follows from Lemma 5.

Conversely, suppose that p MAYF $tt(s)$. We proceed by induction on n such that p MAYF$_n$ $tt(s)$. In the base case $n = 0$, we have that BAD$(tt(s))$. Then by Item (1), we know that $s \in \mathscr{N}^\star$, therefore the result follows from INPUT-BOOMERANG axiom.

In the inductive case $n = m+1$ for some $m \in \mathbb{N}$, the inductive hypothesis states that for every $p' \in A$, $s' \in \mathscr{W}$,

$$p' \text{ MAYF}_m\ tt(s') \text{ implies } p' \stackrel{s'}{\Longrightarrow}. \qquad (IH)$$

We have that $p \mid tt(s) \xrightarrow{\eta} \hat{p} \mid r$ and $\hat{p}$ MAYF$_m$ r for some $\hat{p} \in A$ and $r \in C$. Then we continue by case analysis on the rule used to derive the transition $p \mid tt(s) \xrightarrow{\eta} \hat{p} \mid r$. In the case [PROC], we have that $p \xrightarrow{\tau} \hat{p}$ and $r = tt(s)$. Then (IH) implies that $\hat{p} \stackrel{s}{\Longrightarrow}$, and the conclusion follows from [WT-TAU].

In the case [TEST], we have that $\hat{p} = p$ and $tt(s) \xrightarrow{\tau} r$. Then it follows from Lemma 10 that $s = s_1.a.s_2.\bar{a}.s_3$ and $r = tt(s_1.s_2.s_3)$ for some $s_1, s_2 \in \mathscr{N}^\star$, $a \in \mathscr{N}$ and $s_3 \in \mathscr{W}$. Then (IH) gives that $p \stackrel{s_1.s_2.s_3}{\Longrightarrow}$. Moreover, INPUT-BOOMERANG axiom implies that $p \xrightarrow{a} q \xrightarrow{\bar{a}} p$ for some $q \in A$, therefore it follows from applying Lemma 7(2).

In the case [COM], we have that $p \xrightarrow{\mu} \hat{p}$ and $tt(s) \xrightarrow{\bar{\mu}} r$. Then Lemma 9 implies that $s = s_1.\mu.s_2$ and $r = tt(s_1.s_2)$ for some $s_1 \in \mathscr{N}^\star$ and $s_2 \in \mathscr{W}$. It follows from (IH) that $\hat{p} \stackrel{s_1.s_2}{\Longrightarrow}$, and the conclusion follows from Lemma 7(1).

In the case [DELAY], we have that $p \xrightarrow{d} \hat{p}$ and $tt(s) \xrightarrow{d} r$. Then Lemma 11 gives that $s = s_1.(d+d').s_2$ and $r = tt(s_1.d'.s_2)$ for some $s_1 \in \mathscr{N}^\star$, $d' \in \mathbb{R}_{\geq 0}$ and $s_2 \in \mathscr{W} \setminus (\mathbb{R}_{>0} \cdot \mathscr{W})$. Then (IH) gives that $\hat{p} \stackrel{s_1.d'.s_2}{\Longrightarrow}$, and we conclude by Lemma 7(3). □

We are at last ready to prove the alternative characterisation of the MAY-preorder.

Theorem 1. *For every TLTS $\mathscr{L}_A, \mathscr{L}_B, \mathscr{L}_C$, and $p \in A$, $q \in B$,*

(i) if $p \preccurlyeq_{tr} q$ then $p \sqsubseteq^{\mathscr{L}_C}_{\mathrm{MAYF}} q$;

(ii) if $\mathscr{L}_A, \mathscr{L}_B \models \mathrm{OCM} \wedge \mathrm{FFB} \wedge \mathrm{IB}$ and $\mathscr{L}_C \models$ AXCMPL then $p \sqsubseteq^{\mathscr{L}_C}_{\mathrm{MAYF}} q$ implies $p \preccurlyeq_{tr} q$.

Proof. To prove (i) suppose that $p \preccurlyeq_{tr} q$, and let p MAYF r for some client $r \in C$. We have to show that q MAYF r. First we apply Lemma 4 to unzip a failed computation of $p \mid r$. This gives us $p \stackrel{s}{\Longrightarrow}$ and $r \stackrel{\bar{s}}{\Longrightarrow} r'$ together with BAD(r') for some $s \in \mathscr{W}$ and $r' \in C$. The hypothesis implies that $q \stackrel{s}{\Longrightarrow} q'$ for some q', and an application of Lemma 5 shows that $q \mid r \stackrel{d}{\Longrightarrow} q' \mid r'$ for some delay $d \in \mathbb{R}_{\geq 0}$. We have thus the required q MAYF r, because BAD(r').

To prove (ii) suppose $p \sqsubseteq^{\mathscr{L}_C}_{\mathrm{MAYF}} q$, and let a trace $s \in \mathscr{W}$ such that $p \stackrel{s}{\Longrightarrow}$. We have to show that $q \stackrel{s}{\Longrightarrow}$. Thanks to the assumptions on the TLTS $\mathscr{L}_A, \mathscr{L}_B, \mathscr{L}_C$, we can apply Proposition 1 to p and s, thereby obtaining p MAYF $tt(s)$, and thus the hypothesis $p \sqsubseteq^{\mathscr{L}_C}_{\mathrm{MAYF}} q$ implies that q MAYF $tt(s)$. We conclude via another application of Proposition 1. □

Since OCM, FFB and IB are axioms of FWDAX, and in this class there are enough LTS to satisfy AXCMPL, we have that Theorem 1 holds for TFWD. The asymmetric presentation of the soundness and of the completeness statements is useful to stress the generality of former, and to highlight the role of AXCMPL.

4.1 Proofs of test properties

This is a technical subsection in which we prove the lemmas we have stated and used to prove Theorem 1. They are all properties of the behaviours of tests. A reader interested in the gist of this paper need not study the following proofs.

Lemma 8. *There exists $r \in C$ such that $tt(s) \stackrel{\overline{s}}{\Longrightarrow} r$ and* $\mathrm{BAD}(r)$.

Proof. We proceed by induction on the structure of s. In the base case $s = \varepsilon$, we let $r = tt(\varepsilon)$, then have that $\mathrm{BAD}(tt(\varepsilon))$ by Item (1), and we conclude by [WT-REFL].

In the inductive case $s = \lambda.s'$ for some $\lambda \in (\mathsf{Act} \cup \mathbb{R}_{>0})$ and $s' \in \mathscr{W}$, the inductive hypothesis gives that $tt(s') \stackrel{\overline{s'}}{\Longrightarrow} r$ and $\mathrm{BAD}(r)$ for some $r \in C$. If $\lambda = \mu$ for some $\mu \in \mathsf{Act}$ then Item (3) gives that $tt(\mu.s') \xrightarrow{\overline{\mu}} tt(s')$, thus the conclusion follows from [WT-MU]. Otherwise, $\lambda = d$ for some $d \in \mathbb{R}_{>0}$, then Item (5) implies that $tt(d.s') \xrightarrow{d} tt(s')$, thus the conclusion follows from [WT-TIME]. □

Lemma 9. *For every $\mu \in \mathsf{Act}$, $r \in C$, if $tt(s) \xrightarrow{\mu} r$ then there exist two traces $s_1 \in \mathscr{N}^\star$ and $s_2 \in \mathscr{W}$ such that $s = s_1.\overline{\mu}.s_2$ and $r = tt(s_1.s_2)$.*

Proof. Let $\mu \in \mathsf{Act}$. We proceed by induction on the structure of s. The base case $s = \varepsilon$ is trivial from Item (2).

In the inductive case $s = \lambda.s'$ for some $\lambda \in (\mathsf{Act} \cup \mathbb{R}_{>0})$ and $s' \in \mathscr{W}$, the inductive hypothesis states that for every $r' \in C$,

$$tt(s') \xrightarrow{\mu} r' \text{ implies } \exists s'_1 \in \mathscr{N}^\star, s'_2 \in \mathscr{W}.\ s' = s'_1.\overline{\mu}.s'_2 \text{ and } r' = tt(s'_1.s'_2). \qquad (IH)$$

We continue by case analysis on whether $\lambda \in \mathsf{Act}$ or $\lambda \in \mathbb{R}_{>0}$. The second case is trivial by Item (6).

In the first case, $\lambda = \mu'$ for some $\mu' \in \mathsf{Act}$, then we proceed by case analysis on whether $\mu' \in \mathscr{N}$ or $\overline{\mu'} \in \mathscr{N}$. If $\mu' = \overline{a}$ for some $a \in \mathscr{N}$ then it follows from Item (4) that $\mu = a$ and $r = tt(s')$, thus we choose $s_1 = \varepsilon$ and $s_2 = s'$.

Otherwise, $\mu' = a$ for some $a \in \mathscr{N}$, we have that $tt(a.s') \xrightarrow{\overline{a}} tt(s')$ by Item (3). If $\mu = \overline{a}$ then OUTPUT-DETERMINACY axiom implies that $r = tt(s')$, and we let $s_1 = \varepsilon$ and $s_2 = s'$. Otherwise, OUTPUT-CONFLUENCE axiom implies that $r \xrightarrow{\overline{a}} r_1$ and $tt(s') \xrightarrow{\mu} r_1$ for some $r_1 \in C$. Then (IH) gives that $s' = s'_1.\overline{\mu}.s'_2$ and $r_1 = tt(s'_1.s'_2)$ for some $s'_1 \in \mathscr{N}^\star$ and $s'_2 \in \mathscr{W}$. Since $tt(a.s'_1.s'_2) \xrightarrow{\overline{a}} tt(s'_1.s'_2)$ by Item (3), BACKWARD-OUTPUT-DETERMINACY axiom implies that $r = tt(a.s'_1.s'_2)$, thus we let $s_1 = a.s'_1$ and $s_2 = s'_2$. □

Lemma 10. *For every $r \in C$, if $tt(s) \xrightarrow{\tau} r$ then there exist three traces $s_1, s_2 \in \mathscr{N}^\star$, $s_3 \in \mathscr{W}$, and an action $a \in \mathscr{N}$ such that $s = s_1.a.s_2.\overline{a}.s_3$ and $r = tt(s_1.s_2.s_3)$.*

Proof. We proceed by induction on s. The base case $s = \varepsilon$ is trivial from Item (2).

In the inductive case $s = \mu.s'$ for some $\mu \in \mathsf{Act}$ and $s' \in \mathscr{W}$, the inductive hypothesis states that for every $r' \in C$,

$$tt(s') \xrightarrow{\tau} r' \text{ implies } \exists s'_1, s'_2 \in \mathscr{N}^\star,\ a' \in \mathscr{N},\ s'_3 \in \mathscr{W}.\ s' = s'_1.a'.s'_2.\overline{a'}.s'_3 \text{ and } r' = tt(s'_1.s'_2.s'_3). \quad (IH)$$

Then we continue by case analysis on whether $\mu \in \mathscr{N}$ or $\overline{\mu} \in \mathscr{N}$. If $\mu = \overline{b}$ for some $b \in \mathscr{N}$ then it follows from Item (4) that $tt(\overline{b}.s') \xrightarrow{\tau}$, therefore the conclusion is trivial.

Otherwise, $\mu = b$ for some $b \in \mathscr{N}$, we have that $tt(b.s') \xrightarrow{\overline{b}} tt(s')$ by Item (3). It follows from OUTPUT-TAU axiom that either $r \xrightarrow{\overline{b}} r_1$ and $tt(s') \xrightarrow{\tau} r_1$ for some $r_1 \in C$, or $tt(s') \xrightarrow{b} r$. In the first case, (IH) gives that $s' = s'_1.a'.s'_2.\overline{a'}.s'_3$ and $r_1 = tt(s'_1.s'_2.s'_3)$ for some $s'_1, s'_2 \in \mathscr{N}^\star$, $a' \in \mathscr{N}$ and $s'_3 \in \mathscr{W}$. Since $tt(b.s'_1.s'_2.s'_3) \xrightarrow{\overline{b}} tt(s'_1.s'_2.s'_3)$ by Item (3), BACKWARD-OUTPUT-DETERMINACY axiom implies that $r = tt(b.s'_1.s'_2.s'_3)$, thus we choose $s_1 = b.s'_1$, $s_2 = s'_2$, $a = a'$ and $s_3 = s'_3$. In the second case, Lemma 9 implies that $s' = s'_1.\overline{b}.s'_2$ and $r = tt(s'_1.s'_2)$ for some $s'_1 \in \mathscr{N}^\star$ and $s'_2 \in \mathscr{W}$, thus we let $s_1 = \varepsilon$, $s_2 = s'_1$, $a = b$ and $s_3 = s'_2$.

The inductive case $s = d.s'$ for some $d \in \mathbb{R}_{>0}$ and $s' \in \mathscr{W} \setminus (\mathbb{R}_{>0} \cdot \mathscr{W})$ is trivial by Item (6). □

Lemma 11. *For every $d \in \mathbb{R}_{>0}$, $r \in C$, if $tt(s) \xrightarrow{d} r$ then there exist a trace $s_1 \in \mathscr{N}^\star$, a trace $s_2 \in \mathscr{W} \setminus (\mathbb{R}_{>0} \cdot \mathscr{W})$, and a delay $d' \in \mathbb{R}_{\geq 0}$ such that $s = s_1.(d+d').s_2$ and $r = tt(s_1.d'.s_2)$.*

Proof. Let $d \in \mathbb{R}_{>0}$. We proceed by induction on s. The base case $s = \varepsilon$ is trivial from Item (2).

In the inductive case $s = \mu.s'$ for some $\mu \in \mathsf{Act}$ and $s' \in \mathscr{W}$, the inductive hypothesis states that for every $r' \in C$,

$$tt(s') \xrightarrow{d} r' \text{ implies } \exists s'_1 \in \mathscr{N}^\star, d_1 \in \mathbb{R}_{\geq 0}, s'_2 \in \mathscr{W} \setminus (\mathbb{R}_{>0} \cdot \mathscr{W}).$$
$$s' = s'_1.(d+d_1).s'_2 \text{ and } r' = tt(s'_1.d_1.s'_2). \quad (IH)$$

Then we continue by case analysis on whether $\mu \in \mathscr{N}$ or $\overline{\mu} \in \mathscr{N}$. If $\mu = \overline{a}$ for some $a \in \mathscr{N}$ then it follows from Item (4) that the result is trivial.

Otherwise, $\mu = a$ for some $a \in \mathscr{N}$, we have that $tt(a.s') \xrightarrow{\overline{a}} tt(s')$ by Item (3). Since OUTPUT-CONFLUENCE axiom implies that $r \xrightarrow{\overline{a}} r_1$ and $tt(s') \xrightarrow{d} r_1$ for some $r_1 \in C$. Then (IH) gives that $s' = s'_1.(d+d_1).s'_2$ and $r_1 = tt(s'_1.d_1.s'_2)$ for some $s'_1 \in \mathscr{N}^\star$, $d_1 \in \mathbb{R}_{\geq 0}$ and $s'_2 \in \mathscr{W} \setminus (\mathbb{R}_{>0} \cdot \mathscr{W})$. Since $tt(a.s'_1.d_1.s'_2) \xrightarrow{\overline{a}} tt(s'_1.d_1.s'_2)$ by Item (3), BACKWARD-OUTPUT-DETERMINACY axiom implies that $r = tt(a.s'_1.d_1.s'_2)$, thus we let $s_1 = a.s'_1$, $d' = d_1$ and $s_2 = s'_2$.

In the inductive case $s = \hat{d}.s'$ for some $\hat{d} \in \mathbb{R}_{>0}$ and $s' \in \mathscr{W} \setminus (\mathbb{R}_{>0} \cdot \mathscr{W})$, we have that $0 \leq d \leq \hat{d}$ and $r = tt((\hat{d}-d).s')$ by Item (6). Therefore, we choose $s_1 = \varepsilon$, $d' = \hat{d} - d$ and $s_2 = s'$. □

5 Maximal progress

Now that we have established Theorem 1, we shift our attention to the MAXIMAL-PROGRESS axiom, which is given in Equation (2). This property ensures that τ-transitions have priority over delay transitions, meaning that time is not allowed to elapse whenever an internal action is enabled. This assumption is typical in the context of process algebras for instance [1, Chapter 9] and [39, 49, 24].

Crucially, assuming this MAXIMAL-PROGRESS requires modifying the definition of parallel composition, to ensure that if both p and r enjoy MAXIMAL-PROGRESS so does $p \mid r$. Suppose that p can perform a delay d and an action a, and that r can perform the same delay d and the co-action $\overline{a}$, and that both satisfy maximal progress. Using the (standard) semantics given in Figure 4, applying [DELAY] and

[COM], we prove that $p \mid r$ can perform both a delay transition and an internal transition, thus $p \mid r$ does not satisfy maximal progress.

This issue is addressed by strengthening the [DELAY] rule as follows: In this rule, the added condition ensures that the parallel composition cannot perform any τ-transition during the delay d, therefore enforcing maximal progress.

We now define the operational semantics of the composition $p \mid_{\mathsf{mp}} r$ by induction, via the rules in Figure 4, but replacing [DELAY] with [DELAY-MP]. Lemma 5 does not hold for the operator $- \mid_{\mathsf{mp}} -$.

Example 4. *We show that for some trace s and q, r such that $q \stackrel{s}{\Longrightarrow} q'$ and $r \stackrel{\bar{s}}{\Longrightarrow} r'$ it is* not *true that $q \mid_{mp} r \stackrel{d}{\Longrightarrow} q' \mid_{mp} r'$ for some $d \in \mathbb{R}_{\geq 0}$. Consider the following two TLTS (where non-countable amount of delay transitions due to* TIMEAX *are omitted):*

$$\begin{array}{ccc} q_0 & \xrightarrow{1} & q_1 \\ \downarrow a & & \\ q_2 & & \end{array} \qquad \begin{array}{ccc} r_0 & \xrightarrow{1} & r_1 \\ \downarrow \bar{a} & & \\ r_2 & & \end{array}$$

Both TLTS satisfy trivially the MAXIMAL-PROGRESS *axiom, because they contain no τ transition. Consider the trace $s = 1$.We have that $q_0 \stackrel{s}{\Longrightarrow} q_1$ and $r_0 \stackrel{\bar{s}}{\Longrightarrow} r_1$. Since $q_0 \xrightarrow{a} q_2$ and $r_0 \xrightarrow{\bar{a}} r_2$, rule* [COM] *implies that $q_0 \mid_{mp} r_0 \xrightarrow{\tau} q_2 \mid_{mp} r_2$, then the rule* [DELAY-MP] *cannot be applied, and neither can rules* [PROC] *and* [TEST]. *Therefore, the entire TLTS of the parallel composition $q_0 \mid_{mp} r_0$ is*

$$q_0 \mid_{mp} r_0 \xrightarrow{\tau} q_2 \mid_{mp} r_2$$

and, crucially, we have the required $q_0 \mid_{mp} r_0 \stackrel{d}{\not\Longrightarrow} q_1 \mid_{mp} r_1$ for every $d \in \mathbb{R}_{\geq 0}$. □

Since the proof that the MAY-preorder coincides with trace inclusion depends on Lemma 5 [19, 26], it is natural to wonder whether this standard result holds under the MAXIMAL-PROGRESS axiom, *i.e.* for the operator $- \mid_{\mathsf{mp}} -$. The following example shows that this is not the case.

Example 5 (Trace inclusion is not sound)**.** *Recall the TLTS of q_0 and r_0 discussed in Example 4, and consider the following TLTS, (where the transitions due to* TIMEAX *are omitted):*

$$p_0 \xrightarrow{1} p_1$$

The TLTS of p_0 satisfies the MAXIMAL-PROGRESS *axiom. Since the TLTS of p_0 is a subgraph of the TLTS of q_0, we deduce that $p_0 \preccurlyeq_{tr} q_0$. Suppose that r_1 is the only* bad *state in the client.*

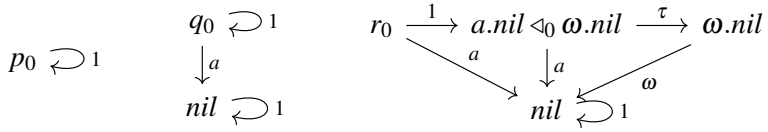
In Example 4 we have already proven that $q_0 \mid_{mp} r_0 \xrightarrow{\tau} q_2 \mid_{mp} r_2$ is the only computation of $q_0 \mid_{mp} r_0$. Since in this computation the client does not reach its only bad state, namely r_1, we have that q_0 MAYF r_0. In contrast, we have that

$$\frac{\forall d' \leq 1, \forall p'', \forall r''.\ p_0 \mid_{mp} r_0 \xrightarrow{d'} p'' \mid_{mp} r'' \stackrel{\tau}{\not\longrightarrow} \qquad p_0 \xrightarrow{1} p_1 \qquad r_0 \xrightarrow{1} r_1}{p_0 \mid_{mp} r_0 \xrightarrow{1} p_1 \mid_{mp} r_1}\ \text{[DELAY-MP]}$$

which proves that p_0 MAYF r_0. This shows that using the semantics of parallel composition that preserves MAXIMAL-PROGRESS, *we have $\preccurlyeq_{tr} \not\subseteq \sqsubseteq_{\text{MAYF}}$ (Equation* (3)*).* □

Recall now Timed CCS [48, 1] and the theory of testing developed for it in [49]. Theorem 1 of *op. cit.* states that the MAY-preorder coincides with language inclusion. We give a counter example this this statement.

Example 6. *Let $p_0 = nil$, $q_0 = a.nil$ and $r_0 = a.nil \triangleleft_1 \omega.nil$ in Timed* CCS. *Their TLTS are as follows (where the transitions due to* TIMEAX *and persistency are omitted for readability):*



Then the only bad state is $\omega.nil$ and exactly the same reasoning as in Example 5 applies. □

We conclude this section remarking the phenomenon shown in the three examples above appears using merely synchronous communication, *i.e.* it does not require using any of the axioms in FWDAX or FDBAX.

6 Related work

Untimed settings with asynchronous communication. The MAY-preorder has been characterised for asynchronous CCS (ACCS), in [17] by comparing traces with so-called *asynchronous* traces. Asynchronous traces correspond to traces of LTSs in the class FWDAX (*i.e.* with forwarding), and our Theorem 1 can be seen as a generalisation to TLTS of their result. Another characterisation for ACCS is given in [12] which relies on a preorder on traces (see Table 2 *op. cit.*). Our Lemma 7 adapts to TLTS statements given in [8, Lemmas 6.10, 6.11, 6.14]. These morphisms on finite words can be traced back to the preorder on traces of [12]. To the best of our knowledge, the most recent characterisation of the MAY-preorder for an asynchronous setting is the one presented in [5]. It relies on a saturated LTS where the transitions added to perform the saturation are exactly those that in FWDAX are due to the axiom INPUT-BOOMERANG.

Timed settings with synchronous communication. The authors of [39] study an augmented version of Timed CSP [46] where delays in syntactic terms are restricted to integer values. They show that some widely used real-time properties, such as safe reachability, constant availability, or bounded response, can be captured as Timed CSP refinement. The notion of "correct implementation" with respect to a specification is defined by trace inclusion (for three safety properties: reachability, bounded invariance and bounded response) and refusal trace inclusion (for two liveness properties: constant availability and timestop-freedom). Moreover, their Theorem 4 shows that using discrete time suffices for their purposes.

Related, several timed semantics have been considered for Timed CSP [38, 46], including early works of Reed and Roscoe [40] and Davies [18]. Schneider's treatment [46], includes a (continuous) timed failures model (TF), and a finite version (TFT), where [46, p. 364] it is observed (without proof) that may testing corresponds to refinement in TFT and must testing to refinement in TF. Moreover in [46, Ch. 13], several (timewise) refinement relations are studied between continuous Timed CSP and untimed and discrete models, enabling, for example, the reuse of CSP's refinement-checker FDR [21], including by mapping to a theory with a special event $tock$ [43], much like the role of σ in [24] and similarly as seen in other process calculi [23, 34, 36], denoting the passage of discrete time. In the *tock*-CSP setting [7], refinement captures both safety and liveness over time.

In [16], the authors characterise the MAY-preorder by trace inclusion for real-time systems modelled as timed transition systems. In their setting, tests specify timing constraints on the discreet actions via resettable clocks, and infinite executions are handled using a Büchi-like notion of trace. Moreover, their labels combine delays and actions into pairs, whereas we treat delays and actions independently. The difference between our work is that we have shown that axioms for asynchronous communication and

for times can be used together. Program communication in [16] instead is synchronous. Moreover our arguments rely on axioms which, for completeness, we had to find, while the authors of [16] work on concrete Büchi-automata.

The authors of [24] characterise the MUST-preorder via an alternative preorder that differs from the standard ones based on acceptance sets or must sets. Their characterisation is developed for a process algebra with discrete time (*i.e.* the passage of time is represented by an action σ), and it accounts for MAXIMAL-PROGRESS.

7 Conclusion

A classical result by Alpern and Schneider [2, 3] is that every property of a program is a conjunction of a safety one and of a liveness one. To ensure that software updates do not cause any regression, we thus need techniques to preserve these two kinds of properties, and in concurrency theory the MAY and MUST preorders do exactly this.

In this paper we have proven that in a setting where programs are timed and communicate via a shared medium, *i.e.* asynchronously, the MAY-preorder is characterised by the inclusion of timed weak traces (Theorem 1). Since we work axiomatically, the proof of completeness required conceiving the novel axioms for tests given in Table 1. Moreover, we have proposed a non-standard definition of weak timed trace that will help us add the presented proofs to our existing Rocq development [10].

Perhaps more importantly, in Section 5 we have shown that if we use the semantics of parallel composition that is compatible with the MAXIMAL-PROGRESS axiom, then both the soundness of trace inclusion *wrt* MAY-preorder, and the standard zipping result (Lemma 5) are false. Since the authors of [49] work in a setting where MAXIMAL-PROGRESS holds ([49, Lemma 2.4]), the counterexamples presented in Section 5 suggest that Theorem 1 and Theorem 2 of [49] warrant a novel careful analysis.

Theorem 3.4 of [24] is a starting point to develop a testing theory for TLTS in presence of MAXIMAL-PROGRESS and continuous time. Moreover adapting it to asynchronous communication will be a test for the approach advocated in [9], *i.e.* just use forwarding. The test axioms in Table 1 provide a basis for deriving the axioms in this setting.

The standard characterisations of the MUST-preorder depend on the notion of weak trace. We claim that Theorem 1 gives the right notion for the setting discussed in this paper, and we are already using it to pursue the study of the MUST-preorder. Whether the notion of barb used in [24] can be adapted to work with our weak timed traces remains to be seen.

An orthogonal issue is whether real numbers are actually necessary to characterise testing preorders for calculi with timeouts. For instance, the authors of [13] require time delays merely to carry a monoid structure. This may be sufficient to formulate our proofs, and we plan to investigate this matter using (axioms capturing the semantics of) both timed CCS and timed CSP.

The rich literature on **ioco**-testing may provide techniques to reason on contextual preorders in timed settings. For instance both [45, Definition 5] and the definition of the relation *tioco* in [6] are reminiscent of the preorder $\preccurlyeq_M$ given in [9, Section 4]. We leave a careful comparison with the literature on **ioco** as future work.

Given the connections between MAY and MUST preorders, and the (extensional) equality of the MUST-preorder with failure divergence refinement ([9, Corollary 3]), we hope that this submission will spur collaboration within the process calculi community to work on joint results on timed CSP, mechanisations, and tools.

References

[1] Luca Aceto, Anna Ingólfsdóttir, Kim Guldstrand Larsen & Jiri Srba (2007): *Reactive Systems: Modelling, Specification and Verification*. Cambridge University Press.

[2] Bowen Alpern & Fred B. Schneider (1985): *Defining Liveness*. *Inf. Process. Lett.*

[3] Bowen Alpern & Fred B. Schneider (1987): *Recognizing Safety and Liveness*. *Distributed Comput.*

[4] Roberto M. Amadio & Giovanni Bernardi (2027): *Operational Methods in Semantics*. WorldScientific.

[5] Paolo Baldan, Filippo Bonchi, Fabio Gadducci & Giacoma Valentina Monreale (2015): *Asynchronous Traces and Open Petri Nets*. In: *Programming Languages with Applications to Biology and Security - Essays Dedicated to Pierpaolo Degano on the Occasion of His 65th Birthday*.

[6] James Baxter, Ana Cavalcanti, Maciej Gazda & Robert M. Hierons (2023): *Testing using CSP Models: Time, Inputs, and Outputs*. *ACM Trans. Comput. Log.*

[7] James Baxter, Pedro Ribeiro & Ana Cavalcanti (2022): *Sound reasoning in tock-CSP*. *Acta Informatica*.

[8] Giovanni Bernardi, Ilaria Castellani, Paul Laforgue, Vincent Padovani & Léo Stefanesco (2026): *Constructive characterisations of the MUST-preorder for asynchrony*. *TOPLAS*. To appear.

[9] Giovanni Bernardi, Ilaria Castellani, Paul Laforgue & Léo Stefanesco (2025): *Constructive characterisations of the MUST-preorder for asynchrony*. In: *ESOP*.

[10] Giovanni Bernardi, Ilaria Castellani, Paul Laforgue & Leo Stefanesco (2025): *Formal proofs for Constructive characterisations of the MUST-preorder for asynchrony*. Available at `https://zenodo.org/records/14735088`.

[11] Giovanni Bernardi, Hugo Férée & Gaëtan Lopez (2026): *A uniform characterisation of the (a)synchronous must-preorder*. Available at `https://hal.science/hal-05602369`.

[12] Michele Boreale, Rocco De Nicola & Rosario Pugliese (2002): *Trace and Testing Equivalence on Asynchronous Processes*. *Inf. Comput.*

[13] Tomasz Brengos & Marco Peressotti (2019): *Behavioural equivalences for timed systems*. *Log. Methods Comput. Sci.*

[14] Stephen D. Brookes (1996): *Full Abstraction for a Shared-Variable Parallel Language*. *Inf. Comput.*

[15] Stephen D. Brookes, C. A. R. Hoare & A. W. Roscoe (1984): *A Theory of Communicating Sequential Processes*. *J. ACM*.

[16] Stefan D Bruda & Chun Dai (2010): *A testing theory for real-time systems*. *Int. Journal of Computers*.

[17] Ilaria Castellani & Matthew Hennessy (1998): *Testing Theories for Asynchronous Languages*. In: *FSTTCS*.

[18] Jim Davies (1993): *Specification and proof in real-time CSP*. Distinguished dissertations in computer science, Cambridge University Press.

[19] Rocco De Nicola & Matthew Hennessy (1984): *Testing Equivalences for Processes*. *Theor. Comput. Sci.*

[20] Wan J. Fokkink (2007): *Modelling Distributed Systems*. Springer.

[21] Thomas Gibson-Robinson, Philip J. Armstrong, Alexandre Boulgakov & A. W. Roscoe (2014): *FDR3 - A Modern Refinement Checker for CSP*. In: *TACAS*.

[22] Rob J. van Glabbeek (2010): *The Coarsest Precongruences Respecting Safety and Liveness Properties*. In: *TCS*.

[23] Jan Friso Groote (1990): *Specification and verification of real time systems in ACP*. In: *IFIP WG6.1 Tenth International Symposium on Protocol Specification, Testing and Verification*.

[24] M. Hennessy & T. Regan (1995): *Process Algebra for Timed Systems*. *Inf. Comput.*

[25] Matthew Hennessy (1982): *Powerdomains and nondeterministic recursive definitions*. In: *International Symposium on Programming*.

[26] Matthew Hennessy (1988): *Algebraic theory of processes*. MIT Press.

[27] Matthew Hennessy (2007): *A distributed Pi-calculus*. Cambridge University Press.

[28] Matthew Hennessy & Gordon D. Plotkin (1979): *Full Abstraction for a Simple Parallel Programming Language*. In: *MFCS*.

[29] Kohei Honda & Mario Tokoro (1991): *An Object Calculus for Asynchronous Communication*. In: *ECOOP*.

[30] Alan Jeffrey & Julian Rathke (2005): *A fully abstract may testing semantics for concurrent objects*. *Theor. Comput. Sci.*

[31] Paris C. Kanellakis & Scott A. Smolka (1990): *CCS Expressions, Finite State Processes, and Three Problems of Equivalence*. *Inf. Comput.*

[32] Vasileios Koutavas & Matthew Hennessy (2011): *A Testing Theory for a Higher-Order Cryptographic Language - (Extended Abstract)*. In: *ESOP*.

[33] Nancy A Lynch & Mark R Tuttle (2026): *An Introduction to Input/Output Automata*. *Form. Asp. Comput.*

[34] Faron Moller & Chris M. N. Tofts (1990): *A Temporal Calculus of Communicating Systems*. In: *CONCUR*.

[35] James H. Morris (1969): *Lambda-calculus models of programming languages*. Ph.D. thesis.

[36] Xavier Nicollin & Joseph Sifakis (1994): *The Algebra of Timed Processes, ATP: Theory and Application*. *Inf. Comput.*

[37] F. Nielson, H. Riis Nielson & C. L. Hankin (1999): *Principles of Program Analysis*. Springer.

[38] Joel Ouaknine (2000): *Discrete analysis of continuous behaviour in real-time concurrent systems*. Ph.D. thesis, University of Oxford UK.

[39] Joël Ouaknine & James Worrell (2003): *Timed CSP = closed timed ε-automata*. *Nordic J. of Computing*.

[40] George M. Reed & A. W. Roscoe (1988): *A Timed Model for Communicating Sequential Processes*. *Theor. Comput. Sci.*

[41] Xavier Rival & Kwangkeun Yi (2020): *Introduction to Static Analysis*. MIT Press.

[42] A. W. Roscoe (1997): *The Theory and Practice of Concurrency*. Prentice Hall.

[43] A. W. Roscoe (2010): *Understanding Concurrent Systems*. Texts in Computer Science, Springer.

[44] Davide Sangiorgi & David Walker (2001): *The Pi-Calculus - a Theory of Mobile Processes*. Cambridge University Press.

[45] Julien Schmaltz & Jan Tretmans (2008): *On Conformance Testing for Timed Systems*. In: *FORMATS*.

[46] Steve Schneider (2000): *Concurrent and Real-time Systems: The CSP Approach*. John Wiley & Sons, Inc.

[47] Peter Selinger (1997): *First-Order Axioms for Asynchrony*. In: *CONCUR*, Springer.

[48] Wang Yi (1991): *CCS + Time = An Interleaving Model for Real Time Systems*. In: *ICALP*.

[49] Shoji Yuen, Toshiki Sakabe & Yasuyoshi Inagaki (1996): *Symbolic Alternative Characterizations of Testing Preorders for Regular Timed Processes*. *RIMS Technical report*.

A Sanity check of our definition of weak timed trace

We denote by $\xrightarrow{\tau}^*$ the reflexive and transitive closure of τ-labelled transitions. The weak transition relation used in [1, Definition 11.8], which we denote $p \overset{\alpha}{\Longrightarrow}_{\text{AILS}} p'$ is defined only on single actions as follows,

- $p \overset{\tau}{\Longrightarrow}_{\text{AILS}} q$ iff $p \xrightarrow{\tau}^* q$,
- $p \overset{\mu}{\Longrightarrow}_{\text{AILS}} q$ iff $p \overset{\tau}{\Longrightarrow}_{\text{AILS}} p_1 \xrightarrow{\mu} p_2 \overset{\tau}{\Longrightarrow}_{\text{AILS}} p$ for some $p_1, p_2 \in A$,
- $p \overset{d}{\Longrightarrow}_{\text{AILS}} q$ iff $p \overset{\tau}{\Longrightarrow}_{\text{AILS}} p_1 \xrightarrow{d_1} q_1 \cdots q_{n-1} \overset{\tau}{\Longrightarrow}_{\text{AILS}} p_n \xrightarrow{d_n} q_n \overset{\tau}{\Longrightarrow}_{\text{AILS}} q$ and $d = \sum_{i=1}^{n} d_i$ for some $n \in \mathbb{N}$, $d_1, \ldots, d_n \in \mathbb{R}_{\geq 0}$, and $p_1, q_1, \ldots, p_n, q_n \in A$.

We lift their weak transition to finite traces $s \in \mathscr{W}$ such that $s \neq \varepsilon$ in the standard way:

$$p \overset{s}{\Longrightarrow}_{\text{AILS}} p' \text{ iff } p \overset{\lambda_1}{\Longrightarrow}_{\text{AILS}} p_1 \dots p_{n-1} \overset{\lambda_n}{\Longrightarrow}_{\text{AILS}} p' \text{ and } s = \lambda_1 \dots \lambda_n \tag{5}$$

for some $n \in \mathbb{N}$, $\lambda_1, \dots, \lambda_n \in (\mathsf{Act} \cup \mathbb{R}_{>0})$, and $p_1, \dots, p_{n-1} \in A$.

Lemma 12. *For every LTS $\mathscr{L}_A$, $p, q \in A$, $s \in \mathscr{W}$, if $s \neq \varepsilon$ then $p \overset{s}{\Longrightarrow} q$ iff $p \overset{s}{\Longrightarrow}_{\text{AILS}} q$.*

Proof. By induction on s using Equation (5) and lemmas 13 and 14. □

Lemma 13. *For every LTS $\mathscr{L}_A$, $p, q \in A$, $d \in \mathbb{R}_{\geq 0}$, $p \overset{d}{\Longrightarrow} q$ if and only if $p \overset{d}{\Longrightarrow}_{\text{AILS}} q$.*

Proof. For the *only if* implication we proceed by induction on $p \overset{d}{\Longrightarrow} q$. The inductive case [WT-MU] is trivial. In the base case [WT-REFL], we have that $d = 0$ and $q = p$, thus $p \overset{0}{\Longrightarrow}_{\text{AILS}} q$ by reflexivity of $\overset{\tau}{\longrightarrow}^*$. In the inductive case [WT-TAU], we have that $p \overset{\tau}{\longrightarrow} p' \overset{d}{\Longrightarrow} q$ for some $p' \in A$ and the inductive hypothesis gives that $p' \overset{d}{\Longrightarrow}_{\text{AILS}} q$. Therefore, $p \overset{d}{\Longrightarrow}_{\text{AILS}} q$ follows by transitivity of $\overset{\tau}{\longrightarrow}^*$. In the inductive case [WT-TIME], we have that $d \in \mathbb{R}_{>0}$, $p \overset{d-d'}{\longrightarrow} p' \overset{d'}{\Longrightarrow} q$ for some $d' \in \mathbb{R}_{\geq 0}$ and $p' \in A$ and the inductive hypothesis gives that $p' \overset{d'}{\Longrightarrow}_{\text{AILS}} q$. Therefore, $p \overset{d}{\Longrightarrow}_{\text{AILS}} q$ follows by reflexivity of $\overset{\tau}{\longrightarrow}^*$.

For the *if* implication, we proceed by induction on $|p \overset{d}{\Longrightarrow}_{\text{AILS}} q|$. In the base case $|p \overset{d}{\Longrightarrow}_{\text{AILS}} q| = 0$, we have that $d = 0$ and $p \overset{\tau}{\longrightarrow}^* q$. Then we continue by induction on $p \overset{\tau}{\longrightarrow}^* q$. In the base case $q = p$ and we conclude by [WT-REFL]. In the inductive case $p \overset{\tau}{\longrightarrow} p' \overset{\tau}{\longrightarrow}^* q$ for some $p' \in A$ and the inductive hypothesis gives that $p' \overset{\varepsilon}{\Longrightarrow} q$. Therefore, we conclude by [WT-TAU]. In the inductive case $|p \overset{d}{\Longrightarrow}_{\text{AILS}} q| = n+1$ for some $n \in \mathbb{N}$, we have that $p \overset{\tau}{\longrightarrow}^* p_1 \overset{d_1}{\longrightarrow} q_1$ and $|q_1 \overset{d-d_1}{\Longrightarrow}_{\text{AILS}} q| = n$ for some $d_1 \in \mathbb{R}_{\geq 0}$ and $p_1, q_1 \in A$ and the inductive hypothesis gives that $q_1 \overset{d-d_1}{\Longrightarrow} q$. Then we continue by induction on $p \overset{\tau}{\longrightarrow}^* p_1$. In the base case $p_1 = p$ and we conclude by [WT-TIME]. In the inductive case $p \overset{\tau}{\longrightarrow} p' \overset{\tau}{\longrightarrow}^* p_1$ for some $p' \in A$ and the inductive hypothesis gives that $p' \overset{d}{\Longrightarrow} q$. Therefore, we conclude by [WT-TAU]. □

Lemma 14. *For every LTS $\mathscr{L}_A$, $p, q \in A$, $\mu \in \mathsf{Act}$, $p \overset{\mu}{\Longrightarrow} q$ if and only if $p \overset{\mu}{\Longrightarrow}_{\text{AILS}} q$.*

Proof. For the *only if* implication we proceed by induction on $p \overset{\mu}{\Longrightarrow} q$. And for the *if* implication, we proceed by induction on $|p \overset{\mu}{\Longrightarrow}_{\text{AILS}} q|$. □

B Properties of traces

Suppose that $\mathscr{L}_A \vDash \text{OCM} \wedge \text{FFB}$, and that $p, p', q \in A$.

Lemma 6 (Feedback)**.** *For every $a \in \mathscr{N}$, $s \in \mathscr{W}$, $p \overset{\overline{a}}{\longrightarrow} p' \overset{a.s}{\Longrightarrow} q$ imply $p \overset{s}{\Longrightarrow} q$.*

Proof. We proceed by rule induction on the reduction $p' \overset{a.s}{\Longrightarrow} q$. The base case [WT-REFL], and the inductive case [WT-TIME] is trivial.

In the inductive case [WT-TAU], we have that $p' \overset{\tau}{\longrightarrow} \hat{p} \overset{a.s}{\Longrightarrow} q$ for some $\hat{p} \in A$, and the inductive hypothesis ensures that for every $p_1 \in A$,

$$p_1 \overset{\overline{a}}{\longrightarrow} \hat{p} \text{ implies } p_1 \overset{s}{\Longrightarrow} q. \tag{IH}$$

Since $p \xrightarrow{\overline{a}} p'$, OUTPUT-COMMUTATIVITY axiom gives that $p \xrightarrow{\tau} q' \xrightarrow{\overline{a}} \hat{p}$ for some $q' \in A$. Therefore, by applying (IH) we deduce that $q' \stackrel{s}{\Longrightarrow} q$, and we conclude by [WT-TAU].

In the inductive case [WT-MU], we have that $p' \xrightarrow{a} \hat{p} \stackrel{s}{\Longrightarrow} q$ for some $\hat{p} \in A$. Since $p \xrightarrow{\overline{a}} p'$, FWD-FEEDBACK axiom implies that either $p \xrightarrow{\tau} \hat{p}$, or $p = \hat{p}$. The first case follows from [WT-TAU] and the second case is immediate. □

Lemma 15. *If $\mathscr{L}_A \vDash$ IB, then for every $s_1 \in \mathscr{N}^\star$, $s_3 \in \mathscr{W}$, $\mu \in \mathsf{Act}$, $p \xrightarrow{\mu} p'$ and $p' \stackrel{s_1.s_3}{\Longrightarrow} q$ imply $p \stackrel{s_1.\mu.s_3}{\Longrightarrow} q$.*

Proof. Let $\mu \in \mathsf{Act}$ and $s_3 \in \mathscr{W}$. We proceed by induction on s_1. In the base case $s_1 = \varepsilon$, we have that $p \xrightarrow{\mu} p' \stackrel{s_3}{\Longrightarrow} q$, and the conclusion follows from [WT-MU].

In the inductive case $s_1 = a.s_1'$ for some $a \in \mathscr{N}$ and $s_1' \in \mathscr{N}^\star$, the inductive hypothesis states that for every $p_1, p_2, q_1 \in A$,

$$p_1 \xrightarrow{\mu} p_2 \text{ and } p_2 \stackrel{s_1'.s_3}{\Longrightarrow} q_1 \text{ imply } p_1 \stackrel{s_1'.\mu.s_3}{\Longrightarrow} q_1. \tag{IH}$$

From INPUT-BOOMERANG axiom, we have that $p \xrightarrow{a} \hat{p} \xrightarrow{\overline{a}} p$ for some $\hat{p} \in A$. Since $p \xrightarrow{\mu} p'$, OUTPUT-COMMUTATIVITY axiom implies that $\hat{p} \xrightarrow{\mu} q' \xrightarrow{\overline{a}} p'$ for some $q' \in A$. It follows from Lemma 6 that $q' \stackrel{s_1'.s_3}{\Longrightarrow} q$, thus from (IH), we obtain that $\hat{p} \stackrel{s_1'.\mu.s_3}{\Longrightarrow} q$, and we conclude by [WT-MU]. □

Lemma 16. *If $\mathscr{L}_A \vDash$ IB, then for every $s_1 \in \mathscr{N}^\star$, $s_3 \in \mathscr{W}$, $d, d' \in \mathbb{R}_{\geq 0}$, $p \xrightarrow{d} p'$ and $p' \stackrel{s_1.d'.s_3}{\Longrightarrow} q$ imply $p \stackrel{s_1.(d+d').s_3}{\Longrightarrow} q$.*

Proof. Let $d, d' \in \mathbb{R}_{\geq 0}$ and $s_2 \in \mathscr{W} \setminus (\mathbb{R}_{>0} \cdot \mathscr{W})$. If $d = 0$ then the result is immediate. Otherwise, $d \in \mathbb{R}_{>0}$ and we proceed by induction on s_1. In the base case $s_1 = \varepsilon$, we have that $p \xrightarrow{d} p' \stackrel{d'.s_2}{\Longrightarrow} q$, and the conclusion follows from [WT-TIME].

In the inductive case $s_1 = a.s_1'$ for some $a \in \mathscr{N}$ and $s_1' \in \mathscr{N}^\star$, the inductive hypothesis states that for every $p_1, p_2, q' \in A$,

$$p_1 \xrightarrow{d} p_2 \text{ and } p_2 \stackrel{s_1'.d'.s_2}{\Longrightarrow} q' \text{ imply } p_1 \stackrel{s_1'.(d+d').s_2}{\Longrightarrow} q'. \tag{IH}$$

From INPUT-BOOMERANG axiom, we have that $p \xrightarrow{a} \hat{p} \xrightarrow{\overline{a}} p$ for some $\hat{p} \in A$. Since $p \xrightarrow{d} p'$, OUTPUT-COMMUTATIVITY axiom implies that $\hat{p} \xrightarrow{d} q' \xrightarrow{\overline{a}} p'$ for some $q' \in A$. It follows from Lemma 6 that $q' \stackrel{s_1'.d'.s_2}{\Longrightarrow} q$, thus from (IH), we obtain that $\hat{p} \stackrel{s_1'.(d+d').s_2}{\Longrightarrow} q$, and we conclude by [WT-MU]. □

Lemma 7 (Trace morphisms)**.** *If $\mathscr{L}_A \models$ IB, then for every $s_1 \in \mathscr{N}^\star$, and $s_3 \in \mathscr{W}$, we have*

1. *for every $\mu \in \mathsf{Act}$, $p \xrightarrow{\mu} p' \stackrel{s_1.s_3}{\Longrightarrow} q$ imply $p \stackrel{s_1.\mu.s_3}{\Longrightarrow} q$.*
2. *for every $a \in \mathscr{N}$, $s_2 \in \mathscr{N}^\star$, $p \stackrel{s_1.s_2.s_3}{\Longrightarrow} p'$ implies $p \stackrel{s_1.a.s_2.\overline{a}.s_3}{\Longrightarrow} p'$.*
3. *for every $d, d' \in \mathbb{R}_{\geq 0}$, $p \xrightarrow{d} p' \stackrel{s_1.d'.s_3}{\Longrightarrow} q$ imply $p \stackrel{s_1.(d+d').s_3}{\Longrightarrow} q$.*

Proof. Point (1) follows from Lemma 15, and point (3) follows from Lemma 16. For point (2), INPUT-BOOMERANG implies that $p \xrightarrow{a} p' \xrightarrow{\overline{a}} p$ for some $p' \in A$, then the result follows by applying twice point (1). □