Heterogeneous and Asynchronous Networks of Timed Systems

José L. Fiadeiro[1] and Antónia Lopes[2]

[1]Dep. of Computer Science, Royal Holloway University of London, UK
jose.fiadeiro@rhul.ac.uk
[2]Dep. of Informatics, Faculty of Sciences, University of Lisbon, Portugal
mal@di.fc.ul.pt

Abstract. We present a component algebra and an associated logic for heterogeneous timed systems. The components of the algebra are asynchronous networks of processes that abstract the behaviour of machines that execute according to the clock granularity of the network node in which they are placed and communicate asynchronously with machines at other nodes. The main novelty of our theory is that not all network nodes need to have the same clock granularity: we investigate conditions under which we can guarantee, a priori, that any interconnections generated at run time through dynamic binding of machines with different clock granularities leads to a consistent orchestration of the whole system. Finally, we investigate which logics can support specifications for this component algebra.

1 Introduction

The software systems that are now operating in cyberspace are best modelled as networks of machines, where each machine performs local computations and can be interconnected dynamically (at run time) to other machines to achieve some goal. Because of the distributed nature of such networks, it does not make sense to assume that all nodes of the network, where machines execute, have the same clock granularity. This means that interconnections can only be established asynchronously and through communication channels (other machines) that can orchestrate the interactions between machines according to the clock granularities of the nodes in which they execute.

In this paper, we put forward a component algebra for such heterogeneous and asynchronous networks of timed systems. Our algebra abstracts the behaviour of timed machines as processes whose traces are generated according to the clock granularity of the network node in which they execute. We define a composition operator through which networks can be interconnected and investigate properties of such networks, namely conditions that guarantee consistency, i.e., that the interconnected processes collectively generate a non-empty behaviour. Finally, we discuss how specifications for this component algebra can be supported through a continuous-semantics metric temporal logic.

Contributions. In [6] we put forward a component and interface algebra for service-oriented computing that is based on asynchronous networks of processes interconnected through communication channels. A first extension of this model for timed systems was presented in [4] based on a homogeneous notion of time – all processes execute according to the same time granularity. The present extension to a heterogeneous setting is not

trivial (which justifies this paper) because, where the algebraic properties of composition in an homogenous-time domain generalise those of the un-timed domain presented in [6], interconnection in a heterogeneous setting is much more involved – indeed, not even always admissible. The main challenges come from (a) the fact that the topological properties of timed traces are more intricate than those of un-timed ones, which requires the definition of a new time-related refinement relation and a new time-related closure operator that does not reduce to the Cantor topology of trace-based domains; and (b) the fact that, in a heterogeneous timed domain, different clock granularities interfere with the way processes need to be coordinated in order to ensure that they can cooperate.

We have also generalised the notion of asynchronous relational network (ARN) proposed in [6] so as to capture a larger class of systems where coordination of interactions takes place among groups of processes, not just between pairs of processes. This extension requires a different algebraic structure for the networks, which is why we moved from graphs to hypergraphs. From a software engineering point of view, this shift enables us to provide a much richer mathematical model where, essential properties of run-time interconnections (such as consistency) can still be formulated and analysed, i.e., the ability for systems to work effectively when interconnected. We provide compositionality results for consistency through criteria that can be checked on processes at design time that guarantee the consistency of interconnections when performed at run time across different clock granularities.

In terms of logics through which the behaviour of machines can be specified or analysed, we abandoned the implicit-time model used in [6], which is not realistic for the class of applications that need to run across heterogeneous time domains, in favour of a metric temporal logic [10]. The challenges here concern the requirements of the component model for topological notions of closure that go beyond the traditional safety-related ones, which led us to adopt a continuous semantics with a new operator that captures the required notion of closure.

Related Work. Several researchers have recently addressed discrete timed systems with heterogeneous clock granularities but the focus has not been on the the development of theories of composability for these systems as we do in this paper. An exception is [11], which studies when the composition of heterogeneous tag machines [2] is sound and complete. However, the notion of composition considered therein is more relaxed than ours (allowing for the delay between events to be modified) and, as a consequence, not appropriate for addressing global properties of systems interconnected at run time as actually implemented, which we do by adopting instead a trace-based model in which composition corresponds to intersection. A trace-based model has also the advantage of abstracting from the specificities of the different classes of automata that can be chosen as models of implementations. Because un-timed networks were investigated in [6] over traces, adopting a similar model for timed ones also allows us to better appreciate the differences between un-timed and timed domains.

Formal clock calculi have also been developed that address heterogeneity, for example [7] in which a synchronous data-flow language is proposed that supports the modelling of multi-periodic systems and the refinement of clock granularities in a way that is similar to what we propose this paper. However, the main focus of such calculi has been on modelling and simulation, not so much on the challenges that heterogene-

ity raises on run-time interconnection of systems and, therefore, they are too specific on aspects that do not directly impact on system properties such as consistency. In fact, to the best of our knowledge, ours is the first model that adopts networks as components of systems and, therefore, addresses (run-time) compositionality at the network level.

Several frameworks have also been proposed for component/service-based software systems that exhibit timed properties, although not in a heterogeneous-time context. Algebraic frameworks such as [5,8,9,13] address global properties similar to consistency, such as compatibility – whether the conversation protocols (modelled as timed automata) followed by the peers in a choreography lead to deadlocks or time conflicts that prevent them from completing (e.g., reaching final states). However, the focus in this context is on the modelling of the (timed) conversation protocols that characterise the global behaviour of a (fixed) number of peers. What we investigate in this paper is, instead, conditions through which we can guarantee that the orchestrations of components, whose interconnection is performed at run time, can work together. This has implications on the properties that are required of networks in order to guarantee consistency. An example is the way time is managed: in choreography, this is done globally for the (fixed) set of peers; in our approach, this needs to be done locally at the level of each process because composition is dynamic through run-time binding to machines that may be executing in platforms where the clock granularity is different.

2 Preliminaries

We start by recalling a few concepts related to traces and their Cantor topology. Given a set A, a *trace* λ over A is an element of A^ω. We denote by $\lambda(i)$ the $(i+1)$-th element of λ. A *segment* π is an element of A^*, the length of which we denote by $|\pi|$. We use $\pi < \lambda$ to mean that the segment π is a prefix of λ. Given $a \in A$, we denote by $\pi \cdot a$ the segment obtained by extending π with a. A *property* Λ over A is a set of traces. For every property Λ, we define $\Lambda^f = \{\pi : \exists \lambda \in \Lambda(\pi < \lambda)\}$ — the segments that are prefixes of traces in Λ, also called the *downward closure* of Λ — and $\bar{\Lambda} = \{\lambda : \forall \pi < \lambda(\pi \in \Lambda^f)\}$ — the traces whose prefixes are in Λ^f, also called the *closure* of Λ. A property Λ is said to be *closed* iff $\Lambda \supseteq \bar{\Lambda}$ (and, hence, $\Lambda = \bar{\Lambda}$).

In our model, traces consist of an infinite sequence of pairs of an instant of time and a set of actions — the actions that are observed at that instant. In order to model networks of systems, we allow sets of actions to be empty: on the one hand, this allows us to model finite behaviours, i.e., systems that stop executing actions after a certain point in time while still part of a network; on the other hand, it allows us to model observations that are triggered by actions performed by components outside the system.

Definition 1 (Timed traces) *Let A be a set (of actions).*

- *A* time sequence *τ is a trace over $\mathbb{R}_{\geq 0}$ such that: $\tau(0) = 0$; $\tau(i) < \tau(i+1)$ for every $i \in \mathbb{N}$; the set $\{\tau(i) : i \in \mathbb{N}\}$ is unbounded, i.e., time progresses (the 'non-Zeno' condition). An* action sequence *σ is a trace over 2^A such that $\sigma(0) = \emptyset$.*
- *A* timed trace *over A is a pair $\lambda = \langle \sigma, \tau \rangle$ of an action and a time sequence. We denote the sets of timed traces and segments over A by $\Lambda(A)$ and $\Pi(A)$, respectively.*

– *Given a timed property $\Lambda \subseteq \Lambda(A)$ we define, for every time sequence τ, $\Lambda_\tau = \{\sigma \in (2^A)^\omega : \langle \sigma, \tau \rangle \in \Lambda\}$ — the action property defined by Λ and τ, and $\Lambda_{time} = \{\tau : \exists \sigma \in (2^A)^\omega (\langle \sigma, \tau \rangle \in \Lambda)\}$ — the time sequences of traces in Λ.*
– *Given $\delta \in \mathbb{R}_{>0}$, the* δ-time sequence *τ_δ is defined by $\tau_\delta(i) = i \cdot \delta$ for every $i \in \mathbb{N}$. A* δ-timed trace *over A is a timed trace $\langle \sigma, \tau_\delta \rangle$, the set of which is denoted by $\Lambda^\delta(A)$. A* δ-timed property *is a timed property that consists of δ-timed traces.*

Definition 2 (Time refinement) *Let $\rho : \mathbb{N} \to \mathbb{N}$ be a monotonically increasing function that satisfies $\rho(0) = 0$.*

– *Let τ, τ' be two time sequences. We say that τ'* refines *τ through ρ, which we denote by $\tau' \preceq_\rho \tau$, iff, for every $i \in \mathbb{N}$, $\tau(i) = \tau'(\rho(i))$. We say that τ'* refines *τ, which we denote by $\tau' \preceq \tau$, iff $\tau' \preceq_\rho \tau$ for some ρ.*
– *Let $\lambda = \langle \sigma, \tau \rangle$, $\lambda' = \langle \sigma', \tau' \rangle$ be two timed traces. We say that λ' refines λ through ρ — which we denote by $\lambda' \preceq_\rho \lambda$ — iff $\tau' \preceq_\rho \tau$ and, for every $i \in \mathbb{N}$ and $\rho(i) < j < \rho(i+1)$, $\sigma(i) = \sigma'(\rho(i))$ and $\sigma'(j) = \emptyset$. We also say that λ' refines λ — which we denote by $\lambda' \preceq \lambda$ — iff $\lambda' \preceq_\rho \lambda$ for some ρ.*
– *The r-closure of a timed property Λ is $\Lambda^r = \{\lambda' : \exists \lambda \in \Lambda (\lambda' \preceq \lambda)\}$. We say that Λ is* closed under time refinement*, or r-closed, iff $\Lambda^r \subseteq \Lambda$.*

A time sequence refines another if the former interleaves time observations between any two time observations of the latter. Refinement extends to traces by requiring that no actions be observed in the finer trace between two consecutive times of the coarser.

It is not difficult to prove that the refinement relation makes the space of all time sequences a complete meet semi-lattice, the meet of two time sequences ρ_1 and ρ_2 being given by the recursion $\rho(i+1) = min(\{\rho_1(j) > \rho(i), j \in \mathbb{N}\} \cup \{\rho_2(j) > \rho(i), j \in \mathbb{N}\})$ together with the base $\rho(0) = 0$. However, if one considers the space of all δ-time sequences $\{\tau_\delta : \delta \in \mathbb{R}_{>0}\}$, it is easy to see that a meet of τ_{δ_1} and τ_{δ_2} exists iff δ_1 and δ_2 are commensurable, i.e., there are $n, m \in \mathbb{N}_{>0}$ such that $\delta_1/n = \delta_2/m$, in which case the meet is τ_δ where δ is their greatest common divisor.

Functions between sets of actions *(alphabet maps)* are useful for defining relationships between individual processes and the networks in which they operate.

Definition 3 (Projection and translation) *Let $f : A \to B$ be a function (alphabet map).*

– *For every $\sigma \in (2^B)^\omega$, we define $\sigma|_f \in (2^A)^\omega$ pointwise as $\sigma|_f(i) = f^{-1}(\sigma(i))$ — the* projection *of σ over A. If f is an inclusion, then we tend to write $\_|_A$ instead of $\_|_f$ (when applied to a trace, $\_|_A$ forgets the actions of B that are not in A).*
– *For every timed trace $\lambda = \langle \sigma, \tau \rangle$ over B, we define its projection over A to be $\lambda|_f = \langle \sigma|_f, \tau \rangle$, and for every timed property Λ over B, $\Lambda|_f = \{\lambda|_f : \lambda \in \Lambda\}$ — the* projection *of Λ to A.*
– *For every timed property Λ over A, we define $f(\Lambda) = \{\langle \sigma, \tau \rangle : \langle \sigma|_f, \tau \rangle \in \Lambda\}$ — the* translation *of Λ to B.*

We are particularly interested in translations defined by prefixing every element of a set with a given symbol. Such translations are useful for identifying in a network the machine to which an action belongs — we do not assume that machines have mutually disjoint alphabets. More precisely, given a set A and a symbol p, we denote by $(p.\_)$ the function that prefixes the elements of A with '$p.$'.

3 Heterogeneous Timed Asynchronous Relational Nets

We put forward a generalisation of the component algebra proposed in [6]. The main differences are that (1) we address networks of processes that operate over heterogeneous time, and (2) we use hypergraphs instead of simple graphs in order to account for multiple, not just peer-to-peer interactions. We start by detailing the communication model and then proceed to defining networks and investigating some of their properties.

3.1 Processes and connections

Our communication model is asynchronous, interactions being based on the exchange of messages. We organise messages in sets that we call ports: a *port* is a finite set (of messages). Ports are communication abstractions that are convenient for organising networks of systems as formalised below. Every message belonging to a port has an associated *polarity*: $^-$ if it is an outgoing message (published at the port) and $^+$ if it is incoming (delivered at the port). Therefore, every port M has a partition $M^- \cup M^+$. For every port M we define its dual M^{op}, which is obtained by swapping the polarities of the messages in M, i.e., $M^{op^-} = M^+$ and $M^{op^+} = M^-$.

The actions of sending (publishing) or receiving (being delivered) a message m are denoted by $m!$ and $m¡$, respectively. More specifically, if M is a port, we define $A_{M^-} = \{m! : m \in M^-\}$, $A_{M^+} = \{m¡ : m \in M^+\}$, and $A_M = A_{M^-} \cup A_{M^+}$ — the set of actions associated with M. Even if a process does not refuse the delivery of messages it can decide to discard them, e.g., if they arrive outside the expected protocol, and not all published messages can be guaranteed to be delivered to their destination.

Definition 4 (Process) *A* process *is a triple $P = \langle \delta, \gamma, \Lambda \rangle$ where: (1) $\delta \in \mathbb{R}_{>0}$ is the granularity of the clock of the process; (2) γ is a finite set of mutually disjoint ports; (3) Λ is the r-closure of a non-empty δ-timed property over $A_\gamma = \bigcup_{M \in \gamma} A_M$ defining the behaviour of the process.*

The fact that processes are r-closed means that they contain all possible interleavings of empty observations, thus capturing their behaviour in any possible environment. This notion of closure can be related to mechanisms that, such as stuttering [1], ensure that components do not constrain their environment.

We designate the process $\langle \delta, \{M\}, \Lambda^\delta(A_M)^r \rangle$ by $\boxdot^\delta_M$. This is a process with a single port M that, at any time that is a multiple of its clock granularity, accepts any set of actions belonging to A_M, which henceforth is named RUN.

Our model of interaction is based on orchestrating the joint behaviour of a collection of parties, each of which defines a process; the same party may engage in different orchestrations. Each such orchestration is performed by another process – the orchestrator – that coordinates the joint behaviour of the other parties. Each party is connected to the orchestrator by what we call an attachment:

Definition 5 (Attachment) *An* attachment *is a triple $\langle C, \xi, P \rangle$ where $C = \langle \delta_C, \gamma_C, \Lambda_C \rangle$ and $P = \langle \delta_P, \gamma_P, \Lambda_P \rangle$ are processes and ξ is an injective map from $M_C \in \gamma_C$ to $M_P \in \gamma_P$ that reverses polarities, i.e., $\xi(M_C^+) \subseteq M_P^-$ and $\xi(M_C^-) \subseteq M_P^+$. An attachment is* well formed *iff δ_P is a multiple of δ_C. We often use ξ to designate the whole attachment (triple) if the source and target processes are clear from the context.*

Notice that ξ translates A_{M_C} to A_{M_P} by switching publications and deliveries, i.e., $\xi(m¡) = \xi(m)!$ for $m \in M_C^+$ and $\xi(m!) = \xi(m)¡$ for $m \in M_C^-$. The condition that δ_P is a multiple of δ_C for the attachment to be 'well formed' reflects the fact that the source needs to be able to 'tick' (deliver and receive messages) in a way that is compatible with the target. Attachments are used for building connections:

Definition 6 (Connection) *A* connection *is a triple $\Xi = \langle C, \gamma_F, \xi \rangle$ where: (1) C is a process $\langle \delta, \gamma, \Lambda \rangle$ – the orchestrator of the connection; (2) $\gamma_F \subseteq \gamma$ consists of the ports that are 'free'; (3) ξ assigns to each $M_i \in \gamma$ a well-formed attachment $\xi_i : M_i \to M_{P_i}$ of C to a process P_i such that, (a) for every $M_i \neq M_j$, $M_{P_i} \neq M_{P_j}$, (b) for every $M_i \in \gamma_F$, $P_i = \boxdot^{\delta}_{M_i^{op}}$ and ξ_i is the identity.*

That is, a connection consists of a process that orchestrates interactions among a number of parties. Those parties are attached to the orchestrator, not directly to each other, thus making communication between parties to be asynchronous. Some of the ports of the orchestrator may be 'free', thus accounting for the ability of the connection to grow at run time by accepting new parties, i.e., connections may be open. Those free ports are attached to RUN. Each port of a party can only be used by at most one attachment, i.e., if a party plays different roles in the same connection, it does so via different ports. Because all the attachments in a connection need to be well formed, the clock granularity of each party P_i needs to be a multiple of that of the orchestrator C. Therefore, not all sets of processes can be interconnected: in order to be part of a connection, they need to have a common divisor.

3.2 Networks

Definition 7 (HT-ARN) *A* heterogenous timed asynchronous relational net *(*HT*-ARN) α consists of:*

- *A finite hypergraph $\langle N, E \rangle$ where N is a non-empty finite set of nodes and E is a finite set of hyperedges – each hyperedge is a non-empty set $\{p_i \mid i{=}1..n\}$ of nodes.*
- *A labelling function that assigns a process $\alpha_p = \langle \delta_p, \gamma_p, \mathcal{A}_p \rangle$ to every node p and a connection $\Xi_c = \langle \alpha_c, \gamma_{cF}, \xi_c \rangle$ to every hyperedge c such that:*
  - *i) For every hyperedge c, Ξ_c defines an onto mapping from γ_c to c.*
  - *ii) Each ξ_{ci} is an attachment of α_c to α_{p_i}.*
  - *iii) If two hyperedges c and d share a node p, then the attachments ξ_c and ξ_d associated with p have different codomains, i.e., attach to different ports of α_p.*

We also define the following sets and mappings:

- *A_α is the alphabet associated with α – the union of the alphabets of the processes that label the nodes translated by prefixing all actions with the corresponding node.*
  - *iv) For every $p \in N$, we denote by ι_p is the function that maps A_{γ_p} to A_α, which prefixes the actions of A_{γ_p} with p.*
  - *iiv) For every $c \in E$, we denote by ι_c is the function that maps A_{γ_c} to A_α. This function is such that, for every $p \in c$, $\iota_c(A_M) = \iota_p(\xi_p(A_M))$ where M is the source port of ξ_p. That is, actions of the orchestrator are translated through ξ_p to the attached process p (reversing polarities) and then according to ι_p.*

– $\Lambda_\alpha = \{\lambda \in \Lambda(A_\alpha) : \forall p \in N \cup E\ (\lambda|_{\iota_p} \in \Lambda_{\alpha_p})\}$

Note that, for every $p{\in}N$, $(\_|_{\iota_p})$ first removes the actions that are not in the language $p.A_p$ and then removes the prefix p, and similarly for every $c{\in}E$. Therefore, the set Λ_α consists of all traces over the alphabet of the HT-ARN that are projected to traces of all its processes and channels:

$$\Lambda_\alpha = \bigcap_{p \in N \cup E} \iota_p(\Lambda_{\alpha_p})$$

We take this set to represent the behaviour of α. That is, the behaviour of the HT-ARN is given by the intersection of the behaviour of the processes at the nodes and the hyperedges (connections) translated to the language of the HT-ARN — this corresponds to what one normally understands as a parallel composition in trace-based models. Notice that, because the free ports of connections are labelled with run processes, only the non-free ports are relevant for the behaviour of the HT-ARN. Further notice that, when applied to a set of traces, the translations effectively open the behaviour of the processes to actions in which they are not involved.

As an example, consider a HT-ARN that models a heterogeneous system in which a bank clerk orchestrates a process that receives credit-requests, a process that gets information on risk from a database of clients, and a process that handles approved credit requests. As depicted in Fig. 1, its hypergraph has nodes $c{:}\mathit{CreditRequest}$, $d{:}\mathit{ClientsDB}$, $m{:}\mathit{CreditMgr}$ and $r{:}\textsc{run}$, and the hyperedge $\{c, d, m, r\}{:}\langle \mathit{Clerk}, \{P_k^4\}, id\rangle$ where:

– $\mathit{CreditRequest}$ is a process that has a single port through which it sends $\mathit{creditReq}$ and accept, and receives $\mathit{approved}$, denied and $\mathit{transferDate}$. This process starts by sending a $\mathit{creditReq}$ and waits ten time units for receiving $\mathit{approved}$ or denied. In the first case it sends accept and waits fifty time units for receiving $\mathit{transferDate}$. The granularity δ_c of its clock is 0.5.
– $\mathit{ClientsDB}$ is a process that has a single port through which it receives $\mathit{getClientRisk}$ and sends $\mathit{clientRiskValue}$ and $\mathit{clientRiskUnknown}$. When the first $\mathit{getClientRisk}$ is delivered, it takes no more than seven time units to publish either $\mathit{clientRiskValue}$ or $\mathit{clientRiskUnknown}$. The granularity of its clock is 0.2.
– $\mathit{CreditMgr}$ has a single port through which it receives $\mathit{processCredit}$ and sends $\mathit{expectedDate}$. When the first $\mathit{processCredit}$ is delivered, it takes no more than four time units to publish $\mathit{expectedDate}$. The granularity of its clock is 0.3.
– Clerk is a process with four ports: $P_k^1, P_k^2, P_k^3, P_k^4$. For instance, in port P_k^1, it receives messages $\mathit{creditReq}$ and accept and sends the messages $\mathit{approved}$, denied and $\mathit{transferDate}$. After the delivery of the first $\mathit{creditReq}$ on P_k^1, it publishes $\mathit{getClientRisk}$ on P_k^2 within five time units; then it waits ten time units for the delivery of $\mathit{clientRiskValue}$ or $\mathit{clientRiskUnknown}$ in the same port. If the risk of the transaction is known, this is enough for making a decision and sending $\mathit{approved}$ or denied in port P_k^1; if not, it publishes $\mathit{getRisk}$ on P_k^4 within five time units and waits twenty time units for the delivery of $\mathit{riskValue}$. After sending $\mathit{approved}$ (if ever), Clerk waits forty time units for the delivery of accept, upon which it sends $\mathit{processCredit}$ on P_k^3 within two time units and waits for $\mathit{expectedDate}$; when this happens, it sends $\mathit{transferDate}$ within two time units on P_k^1. The granularity δ_k of its clock is 0.1.

– RUN is $\boxdot^{0,1}_{P^{4\,op}_k}$ and id is a set with four identity attachments (this is because, for ease of presentation, we have picked the same names in every pair of connected ports).

The fact that the port P^4_k is free is represented in Fig. 1 by the grey shadow.

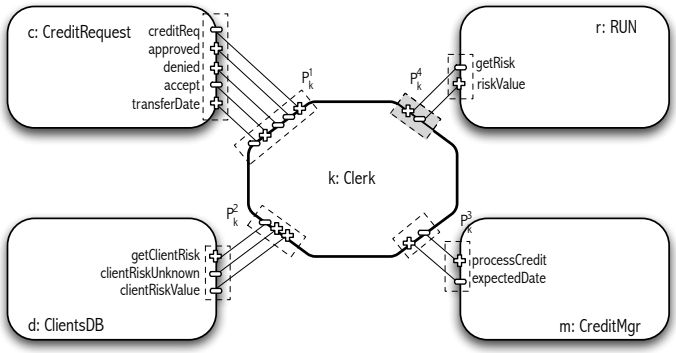

Fig. 1. A HT-ARN consisting of a connection with a free port and three processes.

It is also important to note that, although the existence of a connection between two processes implies that the clock of the orchestrator is a common divisor of those of the processes, it is not necessary that a common divisor exists for all clock granularities of a HT-ARN. However, a common divisor exists for all clock granularities of every sub-net that is connected (i.e., one in which every pair of processes is linked via a path of connections). In particular, if a HT-ARN is a connected hypergraph, this means that it can be implemented over (or simulated by) a single processor.

As in [6], two HT-ARNs can be composed through the ports that are still available for establishing further interconnections, i.e., not connected to any other port, which we call *interaction-points*.

Definition 8 (Interaction-point) *An interaction-point of a* HT-*ARN* α *is a pair* $\langle v, M\rangle$ *such that* $v{\in}N_\alpha$ *is a node and either (1)* $M{\in}\gamma_v$ *is one of its ports and* v *is not attached through* M *to any hyperedge — what we call a* process interaction-point*, or (2)* v *belongs to an hyperedge* $c_v{\in}E_\alpha$*,* M *is a free port of* Ξ_{c_v} *and* v *is attached to* M *— what we call a* connection interaction-point*. We denote by* J_α *the collection of interaction-points of* α*.*

We can interconnect two HT-ARNs by merging process interaction-points with connection interaction-points via attachments that are well-formed:

Definition 9 (Composition of HT-ARNs) *Let* α *and* β *be two* HT-*ARNs with disjoint sets of nodes and* θ *be a family of wires between* α *and* β*, where a wire is a triple*

$$\theta_i = \langle\langle v_i, M_{c_{v_i}}\rangle, \xi_i, \langle p_i, M_{p_i}\rangle\rangle$$

such that, either

1. *$\langle v_i, M_{c_{v_i}}\rangle$ is a connection interaction-point of α, $\langle p_i, M_{p_i}\rangle$ is a process interaction-point of β and $\langle \alpha_{c_{v_i}}, \xi_i : M_{c_{v_i}} \to M_{p_i}, \beta_{p_i}\rangle$ is a well-formed attachment, or*
2. *$\langle v_i, M_{c_{v_i}}\rangle$ is a connection interaction-point of β, $\langle p_i, M_{p_i}\rangle$ is a process interaction-point of α and $\langle \beta_{c_{v_i}}, \xi_i : M_{c_{v_i}} \to M_{p_i}, \alpha_{p_i}\rangle$ is a well-formed attachment,*

with mutually-disjoint sets of interaction points. We define the HT-*ARN $\alpha \|_\theta \beta$ as follows:*

- *Its hypergraph is $\langle N, E\rangle$ where N is obtained from $N_\alpha \cup N_\beta$ by removing the nodes corresponding to the connection interaction-points, and E is obtained from $E_\alpha \cup E_\beta$ by replacing, for each attachment ξ_i:$M_{c_{v_i}} \to M_{p_i}$, the node v_i by p_i in c_{v_i}.*
- *Its node-labelling function γ coincides with γ_α or γ_β on the remaining nodes.*
- *Its hyperedge-labelling function Ξ is as $\Xi_\alpha \cup \Xi_\beta$ except that, for each attachment $\xi_i : M_{c_{v_i}} \to M_{p_i}$, the attachment of the run process at M_{c_v} is replaced with ξ_i and $M_{c_{v_i}}$ is removed from the set of free ports of $\Xi_{c_{v_i}}$.*

In order to illustrate composition of HT-ARNs, consider a HT-ARN that is obtained from the HT-ARN depicted in Fig. 1 by replacing the node c:$CreditRequest$ by r':RUN$'$ and making free the port P_k^1, where RUN$'$ is the process $\boxdot_{P_k^{1\,op}}^{0.1}$. The HT-ARN presented in Fig. 1 is the composition of that HT-ARN and an atomic HT-ARN defined by c:$CreditRequest$ through a wire ξ that connects the interaction-point $\langle r', P_k^1\rangle$ of the first HT-ARN to $\langle c, P_k^{1\,op}\rangle$, the single interaction-point of the latter. The wire ξ is built over the function between the two ports that keeps the names and only reverses polarities. This defines a well-formed attachment because δ_c is a multiple of δ_k.

4 Consistency

The joint consistency of the processes and the orchestrators operating in a HT-ARN is an important property because it ensures that their implementations can work together.

Definition 10 (Consistent HT-ARN) *A* HT-*ARN α is said to be* consistent *if $\Lambda_\alpha \neq \emptyset$.*

In [6], we defined a sub-algebra of (un-timed) ARNs that are consistent and closed under composition. The characterisation of this sub-algebra relied on the closure operator induced by the Cantor topology over action sequences. The same closure operator can be defined over timed traces but, for the purpose of separating the properties required of the action sequences from those of the time sequences and the way they can be checked over automata (which we do in [3]), it is useful to consider other notions of closure.

We can use the Cantor topology over $(2^A)^\omega$ to define a notion of closure relative to a fixed time sequence:

Definition 11 (Closure relative to time) *A timed property Λ is* closed relative to time *or, simply, t-closed, iff, for every $\tau \in \Lambda_{time}$, Λ_τ is closed. A t-closed* HT-*ARN is one in which all processes that label nodes or hyperedges (connections) are t-closed.*

Processes that are closed relative to time define safety properties in the usual un-timed sense: over a fixed time sequence, which cannot be controlled by the processes, the violation of the property can be checked over a finite trace. It is also easy to prove that:

Proposition 12 *Let α be a* HT-*ARN: (a) Λ_α is r-closed; (b) if α is t-closed, so is Λ_α.*

The first follows from the fact that all processes are r-closed by construction. The second follows from the fact that the intersection of t-closed properties is also t-closed.

A property that was found to be relevant in [6] for characterising consistent (untimed) asynchronous relational nets concerns the ability to make joint progress. In the timed version, we need to take into account the way time itself progresses.

Definition 13 (Progress-enabled) *For every* HT-*ARN α and time sequence τ, let*

$$\Pi_{\alpha_\tau} = \{\pi{\in}(2^{A_\alpha})^* : \forall p{\in}N \cup E\ (\pi|_{\iota_p}{\in}\Lambda^f_{\alpha_{p_\tau}})\}$$

We say that α is progress-enabled in relation to τ *iff*

$$\epsilon \in \Pi_{\alpha_\tau} \text{ and } \forall \pi{\in}\Pi_{\alpha_\tau}\ \exists A{\subseteq}A_\alpha((\pi{\cdot}A) \in \Pi_{\alpha_\tau})$$

We say that α is progress-enabled *iff there is a time sequence τ such that α is progress-enabled in relation to every $\tau' \preceq \tau$.*

The set Π_{α_τ} consists of all the segments that the processes can jointly engage in across the time sequence τ. Notice that if Π_{α_τ} is not empty, τ is a refinement of a δ_{α_p}-time sequence for every node or edge p of α. Furthermore, because the intersection of A with the alphabet of any process can be empty, being progress-enabled does not require all parties to actually perform an action.

By itself, being progress-enabled does not guarantee that a HT-ARN is consistent: moving from finite to infinite behaviours requires the analysis of what happens 'at the limit'. However, if we work with t-closed properties, the limit behaviour will remain within the HT-ARN:

Theorem 14 *A* HT-*ARN is consistent if it is t-closed and progress-enabled.*

We now show how HT-ARNs can be guaranteed to be progress-enabled by construction: we identify atomic HT-ARNs that are progress-enabled and prove that the class of progress-enabled HT-ARNs is closed under composition. We start by remarking that, given a process P, the HT-ARN that consists of a single node labelled with P is progress-enabled in relation to at least a δ-time sequence and all its refinements, and therefore is progress-enabled. The same applies to any HT-ARN that consists of a finite set of unconnected processes — in fact, this generalises to any finite juxtaposition of progressed-enabled HT-ARNs (or, indeed, consistent HT-ARNs); the challenge is in checking that progress-enabled HT-ARNs are closed under composition because composition connects HT-ARNs, i.e., it creates connected components.

In [6], we gave criteria for the composition of two (un-timed) progress-enabled ARNs to be progress-enabled based on the ability of processes to buffer incoming messages – being 'delivery-enabled'. In a timed domain, it becomes necessary to identify time sequences across which all parties can work together. Given a HT-ARN and one of its interaction-points $\langle v, M\rangle$, we define the set $D_{\langle v,M\rangle}$ of deliveries that can be made at that point — $D_{\langle v,M\rangle}{=}\{v.m¡ : m{\in}M^+\}$ if $\langle v, M\rangle$ is a process interaction-point and $\{v.m! : m{\in}M^+\}$ otherwise. Notice that in the latter case $\langle v, M\rangle$ is a connection interaction-point and deliveries to that point (in M) are publications by v.

Definition 15 (Delivery-enabled HT-ARN) *A* HT*-ARN* $\alpha = \langle P, C, \delta, \gamma, \Lambda \rangle$ *is delivery-enabled in relation to one of its interaction-points* $\langle v, M \rangle \in I_\alpha$ *if, for every* $B \subseteq D_{\langle v,M \rangle}$, $\tau \in \Lambda_{time}$ *and* $(\pi \cdot A) \in \Pi_{\alpha_\tau}$ *such that* $\tau(|\pi|)$ *is a multiple of* δ_v *(i.e., the process at* v *makes a step),* $(\pi \cdot B \cup (A \setminus D_{\langle v,M \rangle})) \in \Pi_{\alpha_\tau}$ *(i.e., the process at* v *accepts the deliveries in* B *instead of those in* A*.)*

That is, being delivery-enabled at an interaction point requires any joint segment of the HT-ARN over a time sequence to be extensible with any set of messages delivered at that interaction-point. Note that in the case of a connection interaction-point, being delivery-enabled means that the orchestrator of the connection is ready to accept publications at the node v. Also note that being delivery-enabled does not interfere with the decision to publish messages: $B \cup (A \setminus D_{\langle v,M \rangle})$ retains all the publications in A.

Finally, we need to make sure that the processes that orchestrate connections can work together with the processes that they interconnect, i.e., that they do not force the delivery of messages when the processes cannot receive them:

Definition 16 (Cooperative connections) *Let* $\Xi = \langle C, \gamma_F, \xi \rangle$ *be a connection with* $C = \langle \delta, \gamma, \Lambda \rangle$ *and, for every attachment* $\xi_i : M_i \to M_{P_i}$ *of* C *to a process* P_i*, let* $E_i = \{m! : m \in M_i^-\}$*. The connection is said to be* cooperative *if, for every* $\tau \in \Lambda_{time}$ *and for every* ξ_i*, if* $(\pi \cdot A) \in \Lambda^f_\tau$ *and* $\tau(|\pi|)$ *is not a multiple of* δ_{P_i} *then* $\pi \cdot (A \setminus E_i) \in \Lambda^f_\tau$*.*

That is, if after π the connection wants to make a delivery when a process is not in sync, there is an alternative path from π where no delivery is made at that time. Notice that, because δ_{P_i} is a multiple of δ, publications are always made in sync with the orchestrator. Therefore, in the context of a delivery-enabled HT-ARN, if $\tau(|\pi|)$ is not a multiple of δ_{P_i}, $\pi \cdot \emptyset \in \Lambda^f_\tau$.

Theorem 17 *Let* α *be a composition of progress-enabled* HT*-ARNs through a family of wires with mutually-disjoint sets of interaction points i.e.,*

$$\alpha = (\alpha_1 \; \Big|\Big|^{i=1\ldots n}_{\langle\langle v_i, M_{c_{v_i}} \rangle, \xi_i, \langle p_i, M_{p_i} \rangle\rangle} \; \alpha_2)$$

where each $\langle\langle v_i, M_{c_{v_i}} \rangle, \xi_i, \langle p_i, M_{p_i} \rangle\rangle$ *is a wire between* α_1 *and* α_2*. If the connections involved in* θ *(those that label the hyperedges* c_{v_i}*) are cooperative and the* HT*-ARNs are delivery-enabled in relation to the interaction-points being connected, then* α *is progress-enabled.*

Therefore, the proof that a HT-ARN is progress-enabled can be reduced to checking that individual processes and orchestrators are delivery-enabled in relation to their interaction points. To guarantee that the HT-ARN is consistent, it is sufficient to choose processes and orchestrators that are t-closed (implement safety properties). All the checking can be done at design time, not at composition time.

5 A compositional theory for HT-ARNs

In this section, we discuss a logic that supports the specification of timed properties as defined in Section 2 and defines a specification theory for our component algebra.

Several extensions of LTL have been proposed to express and reason about real time, among which Metric Temporal Logic (MTL)[10]. MTL works over timed traces and has been studied extensively in relation to important properties such as decidability. The formulas of MTL are built from a set of atomic propositions A using Boolean connectives and time-constrained versions of the until operator of the form $\mathcal{U}_I$ where $I \subseteq [0, \infty)$ is an interval with endpoints in $\mathbb{Q}_{\geq 0} \cup \{\infty\}$.

$$\phi ::= a \mid \neg\phi \mid \phi \supset \phi \mid \phi \,\mathcal{U}_I\, \phi$$

Our purpose is to be able to use such a logic to define a process through a collection Φ of sentences over the language A_γ of the actions defined by the set γ of process ports, such that the behaviour of the process can be defined as $\{\lambda : \lambda \vDash \Phi\}$, i.e., the set of timed traces that satisfy all the sentences in Φ. We then want to use the inference mechanisms of the logic to be able to derive properties of processes and of HT-ARNs.

Because the behaviour of a process is the r-closure of a non-empty δ-timed property (Def. 4), where δ is a clock granularity, we need to be able to define the process semantics of a collection Φ of sentences in such a way that it meets those requirements. In addition, because only t-closed processes can guarantee good properties of HT-ARNs such as consistency, we should restrict ourselves to a safety fragment.

Fragments of MTL have been characterised in which only safety properties can be expressed such as SAFETY-MTL[12], which requires that sentences are in negation normal form and all eventualities to be time-bounded:

$$\phi ::= a \mid \neg a \mid \phi \wedge \phi \mid \phi \vee \phi \mid \phi \,\mathcal{U}_I\, \phi \mid \phi \,\mathcal{R}_I\, \phi \mid \phi \,\mathcal{R}\, \phi$$

where I is bounded and $\mathcal{R}_I$ (resp. $\mathcal{R}$) is the dual of $\mathcal{U}_I$ (resp. $\mathcal{U}_{[0,\infty)}$) operator. In a time context, a safety property Λ is one that is *divergent safe*, i.e., for any timed trace λ, if for all $\pi{<}\lambda$ there is $\lambda'{\in}\Lambda$ such that $\pi{<}\lambda'$, then $\lambda{\in}\Lambda$. It is easy to see that divergent-safe properties are also t-closed, showing that SAFETY-MTL is adequate for our purposes.

However, the need to specify and reason about r-closed sets of timed traces requires the characterisation of an appropriate fragment of SAFETY-MTL. An alternative, which we take in this paper, is to adopt instead the continuous semantics of MTL. Although the continuous semantics renders MTL and SAFETY-MTL undecidable, it provides a much simpler specification logic for HT-ARNs. We are currently working on the identification of a suitable fragment of SAFETY-MTL with the pointwise semantics.

The models of a continuous semantics are expressed in terms of signals:

Definition 18 (Signal) *A* signal *for an alphabet A is a function $f : \mathbb{R}_{>0} \to 2^A$ with finite variability, that is, with only finitely many discontinuities in any finite amount of time. The semantics of* MTL *over signals is as follows:*

- $f, t \vDash a$ *iff* $a{\in}f(t)$
- $f, t \vDash \neg\phi$ *iff* $f, t \nvDash \phi$
- $f, t \vDash \phi_1 \supset \phi_2$ *iff if* $f, t \vDash \phi_1$ *then* $f, t \vDash \phi_2$
- $f, t \vDash \phi_1 \,\mathcal{U}_I\, \phi_2$ *iff there exists* $u \geq t$ *s.t.* $(u - t) \in I$, $f, u \vDash \phi_2$ *and, for all* $t < r < u$, $f, r \vDash \phi_1$
- $f \vDash \phi$ *iff* $f, 0 \vDash \phi$

Definition 19 (Signals vs timed traces) *Given an alphabet A: (1) a timed-trace $\lambda = \langle \sigma, \tau \rangle$ defines the signal f_λ where, for every i, $f_\lambda(\tau(i)) = \sigma(i)$ and $f_\lambda(t) = \emptyset$ everywhere else; (2) a signal f and a time sequence τ define a timed trace $\lambda_f^\tau = \langle \sigma, \tau \rangle$ where $\sigma(i) = f(\tau(i))$. We use λ_f^δ to denote the δ-timed trace defined by f and τ_δ.*

An important result is that all refinements of a given trace define the same signal:

Proposition 20 *Given timed-traces λ and λ', $\lambda' \preceq \lambda$ implies $f_{\lambda'} = f_\lambda$. It follows that, for every Φ, $\Lambda_\Phi = \{\lambda : f_\lambda \vDash \Phi\}$ is r-closed.*

This is why the continuous semantics provides a 'natural' specification logic for HT-ARNs: only r-closed properties can be specified.

Definition 21 (Process specification) *A specification of a process $\langle \delta, \gamma, \Lambda \rangle$ is $\langle A_\gamma, \Phi \rangle$ such that Φ is in* SAFETY-*MTL and $\Lambda \vDash \Phi$, i.e., for every $\lambda \in \Lambda$, $f_\lambda \vDash \Phi$.*

As an example consider again the process $CreditMgr$ and suppose that its set of timed traces Λ_m is the r-closure of the set of 0.3-timed traces $\langle \sigma, \tau \rangle$ satisfying

$$\forall_{i \in \mathbb{N}} \, (processCredit_{¡} \in \sigma(i) \wedge \forall_{j<i} \, processCredit_{¡} \notin \sigma(j)) \Rightarrow \\ \exists_{k>i} \, (expectedDate! \in \sigma(k) \wedge \tau(k) - \tau(i) < 4 \wedge \forall_{j \neq k} expectedDate! \notin \sigma(j))$$

It is not difficult to prove that, for every $\lambda \in \Lambda_m$,

$$f_\lambda \vDash processCredit_{¡} \mathcal{R} (\neg processCredit_{¡} \vee \Diamond_{<4} expectedDate!)$$

where $\Diamond_{<t} \, \phi$ abbreviates $(true \, \mathcal{U}_{[0,t)} \, \phi)$. This sentence specifies that $expectedDate$ is published within four time units from the first delivery of $processCredit$.

Given now a clock granularity δ and a specification $\langle A_\gamma, \Phi \rangle$, we are interested to know if there is actually a process $\langle \delta, \gamma, \Lambda \rangle$ that it specifies, i.e., if $\langle A_\gamma, \Phi \rangle$ is 'δ-satisfiable'. Note that, because the set of processes that $\langle A_\gamma, \Phi \rangle$ specifies is closed under union, if the set is not empty it will admit a biggest process.

Consider $\Lambda_\Phi = \{\lambda : f_\lambda \vDash \Phi\}$. By Prop. 20, Λ_Φ is r-closed. However, Λ_Φ is not necessarily the r-closure of a set of δ-timed traces. Consider instead the set $\{\lambda : \lambda \in \Lambda^\delta(A_\gamma)$ and $f_\lambda \vDash \Phi\}^r$. To determine if the set is not empty, we would have to find a δ-timed trace λ such that $f_\lambda \vDash \Phi$. For that purpose, we could consider the δ-timed trace λ_g^δ for some $g \vDash \Phi$ (assuming that Φ is logically satisfiable). However, it is not immediate that $f_{\lambda_g^\delta} \vDash \Phi$. This is because g and $f_{\lambda_g^\delta}$ are not necessarily the same signal: λ_g^δ retains only the observations made at multiples of δ and $f_{\lambda_g^\delta}$ then constructs a signal that observes the empty set of actions at all other instants, which g may fail to do. Our approach is to construct a sentence $\mathbb{A}x_\delta$ such that $g \vDash \mathbb{A}x_\delta$ implies $g = f_{\lambda_g^\delta}$. We can then take the set $\Lambda_\Phi^\delta = \{\lambda_f^\delta : f \vDash \Phi$ and $f \vDash \mathbb{A}x_\delta\}^r$ and reduce the δ-consistency of $\langle A_\gamma, \Phi \rangle$ to the satisfiability of $\Phi \cup \{\mathbb{A}x_\delta\}$.

Proposition 22 $\Lambda_\Phi^\delta \vDash \Phi$.

Hence, if $\Phi \cup \{\mathbb{A}x_\delta\}$ is satisfiable, $\langle \delta, \gamma, \Lambda_\Phi^\delta \rangle$ is a process, actually the biggest process that is specified by $\langle A_\gamma, \Phi \rangle$, which we take as its denotation.

We detail now the construction of $\mathbb{A}x_\delta$. We introduce a new class of unary operators $\boxed{\delta}$ where $\delta \in \mathbb{Q}_{>0}$, which allow us to express that a sentence holds at all multiples of δ:

$$f, t \vDash \boxed{\delta}\,\phi \text{ iff for all } n \in \mathbb{N},\ f, t + n \cdot \delta \vDash \phi$$

Notice that restricting δ to $\mathbb{Q}_{>0}$ is not a real limitation. On the one hand, a connected HT-ARN is such that all the clock granularities are commensurate, which means that we can convert them to rational numbers by dividing them by a common divisor. On the other hand, reasoning about HT-ARNs that are not connected is not relevant because disconnected components do not interfere with each other. Notice that, for r-closure, one simply needs a dense set of time granularities.

In the extended language, the sentence $\boxed{\delta}(\Box_{<\delta} \wedge_{a \in A} \neg a)$ — where $\Box_{<t}\ \phi$ is an abbreviation of $\neg(true\, \mathcal{U}_{[0,t)}\ \neg\phi)$ — expresses a key property of δ-timed traces: empty observations occur at all time instants that are not multiple of δ. We denote this sentence by $\mathbb{A}x_\delta$ and, more generally, given $B \subseteq A$, we use $\mathbb{A}x_\delta^B$ to denote $\boxed{\delta}(\Box_{<\delta} \wedge_{a \in B} \neg a)$.

Proposition 23 *$f_\lambda \vDash \mathbb{A}x_\delta$ if λ refines a δ-timed trace.*

Note that, because $\mathbb{A}x_\delta$ is a safety property, we can conclude that Λ_Φ^δ is safe if we restrict Φ to SAFETY-MTL.

We are now interested in reasoning about properties of HT-ARNs. That is, given a HT-ARN α and a sentence ϕ in the language of A_α, we are interested in determining whether $\Lambda_\alpha \vDash \phi$, i.e., $f_\lambda \vDash \phi$ for every $\lambda \in \Lambda_\alpha$.

Theorem 24 *Let α be a HT-ARN and, for every node (resp. hyperedge) p, let Φ_{α_p} be a specification of the process (resp. orchestrator) at p. Let*

$$\Phi_\alpha = \bigcup_{p \in N \cup E} \iota_p(\Phi_{\alpha_p} \cup \mathbb{A}x_{\delta_{\alpha_p}}^{A_{\alpha_p}})$$

We have that $\Lambda_\alpha \vDash \phi$ if $\Phi_\alpha \vdash \phi$.

That is, to prove that ϕ expresses a property of α, it is sufficient to derive ϕ from specifications of the processes and orchestrators of α enriched with the corresponding $\mathbb{A}x_\delta$ axioms.

6 Concluding remarks

In this paper, we have proposed a component algebra that extends the notion of asynchronous relational net developed in [6] to a wider class of systems that operate in a heterogeneous time domain: a HT-ARN is a multigraph of nodes, each with its own clock granularity, where processes execute, and hyperedges where interactions among sets of such processes are orchestrated. Every hyperedge also has its one clock granularity, which needs to be a divisor of the clock granularities of the nodes that it connects so that they can interact. This is important for modelling the software systems that are now starting to operate in cyberspace, where they can connect dynamically, i.e., at run time, to other systems. We provided compositionality results for ensuring the consistency of interconnections when performed at run time across different clock granularities. Contrarily to techniques that operate at design time (e.g., [2]), our results do not require

changes to be performed on the processes that execute in such systems so that they can be interconnected, which would defeat the purpose of supporting dynamic binding.

Our algebra is based on timed traces, which allows us to abstract from the specificities of the different classes of automata that can be chosen as models of implementations and characterise at a higher level the topological properties of the languages generated by such automata that support our compositionality results. In a companion paper [3] we investigate a specific automata-based model of machines, which we intend to extend to networks of automata. Another area of further work concerns the logics that can support an interface algebra for HT-ARNs. Although we provided a version of SAFETY-MTL that can support the specification of HT-ARNs, we had to rely on a continuous semantics to enforce the required closure properties. The problem here is that MTL with a continuous semantics is undecidable; better decidability properties can be obtained by choosing instead a pointwise semantics (i.e., where the logic is interpreted directly over timed traces) [12]. Initial results suggest that a pointwise semantics can be developed for HT-ARNs, though at the cost of restricting the syntax. This is an area in which we are currently working, also capitalising on the automata-based models of processes that we have developed in [3].

References

1. M. Abadi and L. Lamport. The existence of refinement mappings. *Theor. Comput. Sci.*, 82(2):253–284, 1991.
2. A. Benveniste, B. Caillaud, L. P. Carloni, and A. L. Sangiovanni-Vincentelli. Tag machines. In *EMSOFT*, pages 255–263. ACM, 2005.
3. B. Delahaye, J. L. Fiadeiro, A. Legay, and A. Lopes. Heterogeneous timed machines. Technical report, submitted, October 2013.
4. B. Delahaye, J. L. Fiadeiro, A. Legay, and A. Lopes. A timed component algebra for services. In D. Beyer and M. Boreale, editors, *FORTE*, volume 7892 of *LNCS*, pages 242–257. Springer, 2013.
5. G. Díaz, J. J. Pardo, M.-E. Cambronero, V. Valero, and F. Cuartero. Verification of web services with timed automata. *Electr. Notes Theor. Comput. Sci.*, 157(2):19–34, 2006.
6. J. L. Fiadeiro and A. Lopes. An interface theory for service-oriented design. *Theor. Comput. Sci.*, 503:1–30, 2013.
7. J. Forget, F. Boniol, D. Lesens, and C. Pagetti. A multi-periodic synchronous data-flow language. In *HASE*, pages 251–260. IEEE Computer Society, 2008.
8. N. Guermouche and C. Godart. Timed model checking based approach for web services analysis. In *ICWS*, pages 213–221. IEEE, 2009.
9. R. Kazhamiakin, P. K. Pandya, and M. Pistore. Representation, verification, and computation of timed properties in web. In *ICWS*, pages 497–504. IEEE Computer Society, 2006.
10. R. Koymans. Specifying real-time properties with metric temporal logic. *Real-Time Systems*, 2(4):255–299, 1990.
11. T. T. H. Le, R. Passerone, U. Fahrenberg, and A. Legay. Tag machines for modeling heterogeneous systems. In *ACSD*, pages 186–195. IEEE Computer Society, 2013.
12. J. Ouaknine and J. Worrell. Safety metric temporal logic is fully decidable. In *TACAS*, volume 3920 of *LNCS*, pages 411–425. Springer, 2006.
13. J. Ponge, B. Benatallah, F. Casati, and F. Toumani. Analysis and applications of timed service protocols. *ACM Trans. Softw. Eng. Methodol.*, 19(4), 2010.