Under consideration for publication in Math. Struct. in Comp. Science

Dynamic Networks of Heterogeneous Timed Machines

José Fiadeiro[1], Antónia Lopes[2], Benoît Delahaye[3], and Axel Legay[4]

[1] *Department of Computer Science, Royal Holloway University of London, UK*

[2] *Departamento de Informática, Faculdade de Ciências, Universidade de Lisboa, Portugal*

[3] *Université de Nantes / LINA, France*

[4] *INRIA/IRISA, Rennes, France*

Received 8 March 2017

We present an algebra of discrete timed input/output automata that may execute in the context of different clock granularities – which we call timed machines; this algebra includes a refinement operator through which a machine can be extended with new states and transitions in order to accommodate a finer clock granularity as required to interoperate with other machines, and an extension of the traditional product of timed input-output automata to the situation in which the granularities of the two machines are not the same. Over this algebra, we then define an algebra of networks of timed machines that includes operations through which networks can be modified at run time, thus offering a model for systems of interconnected components that can dynamically bind to other systems and, therefore, cannot be adjusted at design time to ensure that they operate in a timed homogeneous setting. We investigate important properties of timed machines such as consistency – in the sense that a machine can be ensured to generate a non-empty language, and feasibility – in the sense that a machine can be ensured to generate a non-empty language no matter what inputs it receives, and propose techniques for checking if timed machines are consistent or are feasible. We generalise those properties to networks of timed machines, and investigate how consistency and feasibility of networks can be proved through properties that can be checked at design time without having to compute, at run time, the product of the machines that operate on those networks, which would not be practical.

1. Introduction

Many software applications operating in cyberspace need to connect, dynamically, to other software systems. For example, in the domain of intelligent transportation, systems for congestion avoidance or coordination of self-driven convoys of cars need to be able to accommodate interconnections that are established at run time between components in ways that cannot be pre-determined at design time.

Applications such as these often have real-time requirements, i.e., their correctness depends not only on what outputs are returned to given inputs, but also on the time at which inputs are received and corresponding outputs are produced and communicated. When components of such software applications, usually written in a high-level programming language and relying on particular time abstractions, are executed in a given execution platform, their real-time behaviour is additionally restricted by the clock period of that platform. Components interconnected at run time will be likely to operate over different clock periods, resulting in a timed heterogeneous system.

Existing formalisms for modelling time-constrained systems focus mainly on mono-periodic systems, i.e., they assume that all system components will operate over a shared clock period. These models can still be used for timed heterogeneous systems whose structure is fixed and known *a-priori* by modelling the system components in terms of a global clock that is the least common multiple of all local clocks. In the case of systems whose structure is dynamic, i.e., modifiable at run time, this is no longer possible (Broy and Stølen, 2001). Based on this, we proposed in (Fiadeiro and Lopes, 2017) a trace-based component algebra and an associated logic for such dynamic heterogeneous timed systems. To the best of our knowledge, no other component algebra has been put forward for timed heterogeneous systems that does not require *a-priori* knowledge of their structure.

In this paper, we investigate another component algebra, which is based on automata like most other models for timed systems (see Sec. 6 for an overview). This is justified by the fact that automata-based models are much closer to implementations than trace-based ones, which therefore allows us to propose modelling abstractions and analyse properties that are more 'operational'. At the same time, working over automata raises challenges that are abstracted away when only the traces that they generate are considered.

More specifically, our model is based on input/output automata and supports *run-time compositionality* in the following sense: it is possible to ensure that components can work together as interconnected over heterogeneous local clocks by relying only on properties of models of those components, with no need for calculating their composition. This is important because calculating and reasoning about the composition at run time is simply not practical, and modifying their time domains at run time to ensure compatibility is not possible.

We distinguish between building networks of timed machines in the sense of interconnecting them so that they interact, and calculating the composition (technically, a fibred product) of timed machines, which is an operation that returns a timed machine. More specifically, we define two algebras: an algebra of timed machines, and an algebra of networks of timed machines; the former is in the tradition of process algebras and addresses component structure, whereas the latter addresses the structure of modern systems, which is typically distributed and dynamic. In particular, we show that the behaviour of a system of interconnected timed machines is not necessarily that of their composition, and investigate when a timed machine exists that offers a best approximation of the behaviour of a given network, which is important for validation of properties and simulation.

Our starting point is the homogeneous timed approach that we proposed in (Delahaye et al., 2013) for services. The extension from a homogenous to a heterogeneous setting is not trivial (which justifies this paper) because, where the algebraic properties of composition in a homogenous-time domain generalise those of an un-timed domain, interconnection in a heterogeneous setting is not even always admissible. For that reason, the model that we propose in Sec. 3 separates the space of discrete timed input/output automata (TIOA) (David et al., 2010; Kaynar et al., 2006) from that of their executions over a given clock: the components of our algebra are pairs of a TIOA and a clock granularity, what we call *timed machines*. Two operations are defined over timed machines: *heterogeneous product*, which extends the traditional product of TIOA to the situation in which the granularities of the two machines are not the same, and *refinement*, which extends a machine with new states and transitions in order to accommodate a finer clock granularity as required to interoperate with other machines.

In Sec. 4, we study two important properties of timed machines: *consistency*, in the sense that a machine can be ensured to generate a non-empty language, and *feasibility*, in the sense that a machine can be ensured to generate a non-empty language no matter what inputs it receives. We prove two compositionality results, one for consistency and the other for feasibility. Those results rely on a number of properties that can be checked, at design time, over given timed machines to ensure that their interconnection will be consistent or feasible without actually having to calculate the product of the corresponding automata at run time.

Finally, in Sec. 5, we extend those results to networks of interfaced timed machines, which provide a model for more complex and dynamic systems. Such networks are finite undirected (multi)graphs whose nodes are labelled with timed machines endowed with interfaces (ports for interconnections) and whose edges are labelled with attachments between ports. Operations for constructing networks are defined, including the binding of a network to another network to create a more complex system. A relationship between the timed-machine and the network algebras is investigated in terms of a notion of 'best approximation' through which we can characterise classes of timed machines that can be used to reason about or simulate networks of timed machines with commensurable clock granularities. Again, we investigate properties through which the consistency and feasibility of such networks can be proved compositionally, i.e., without having to calculate explicitly the composition of the underlying automata. Those properties can be proved at design time and ensure that components that implement the timed machines can work together across different clock granularities.

This paper is an extended version of (Delahaye et al., 2014), which was considerably restructured to accommodate networks and their dynamic behaviour: in addition to Sec. 5, which is totally new, Sec. 3 and Sec. 4 have been revised to accommodate run-time composition. More explanations have also been added throughout the paper, the comparison with related work has been extended, and the proofs of our main results have been included in an appendix.

2. Preliminaries

2.1. *Timed traces*

Although transition systems are typically used as operational semantics of automata (including timed transitions systems for timed automata as in (Henzinger et al., 1991)), we use instead a trace semantics because the topological properties of trace domains allow us to provide a finer characterisation of properties such as consistency and feasibility (cf. Sec. 4). For example, existing transition-system semantics such as (David et al., 2010) offer a weaker notion of consistency for timed automata because it fails to enforce time progression and, therefore, an automaton that does not accept any non-Zeno timed sequence can still be consistent.

We start by recalling a few concepts related to traces. Given a set A, a *trace* λ over A is an element of A^ω, i.e., an infinite sequence of elements of A. In our timed model, we work with *timed traces*, i.e., traces over a cartesian product $2^A \times \mathbb{R}_{\geq 0}$ where A is a set of actions. That is, a timed trace consists of an infinite sequence of pairs of an instant of time and of the set of actions that are observed at that instant of time. Every such set of actions can be empty so that, on the one hand, we can model components that stop executing after a certain point in time while still part of a system and, on the other hand, we can model observations that are triggered by actions performed by components outside the system.

Definition 2.1 (Timed trace and property). Let A be a set (of actions).

— A *time sequence* τ is a trace over $\mathbb{R}_{\geq 0}$ such that:
  – $\tau(0) = 0$;
  – for every $i \in \mathbb{N}$, $\tau(i) < \tau(i+1)$;
  – the set $\{\tau(i) : i \in \mathbb{N}\}$ is unbounded, i.e., time diverges — what is usually called the 'non-Zeno' condition.

— An *action sequence* σ is a trace over 2^A, i.e., an infinite sequence of sets of actions, 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 by $\Lambda(A)$ the set of timed traces over A and by $\Pi(A)$ the set of prefixes of $\Lambda(A)$.

— Given $\delta \in \mathbb{R}_{>0}$, the *$\delta$-time sequence* τ_δ consists of all multiples of δ — i.e., for every $i \in \mathbb{N}$, $\tau_\delta(i) = i \cdot \delta$. A *$\delta$-timed trace* over A is a timed trace $\langle \sigma, \tau_\delta \rangle$.

— A *timed property* over A is a set $\Lambda \subseteq \Lambda(A)$.

That is, in δ-timed traces, actions occur according to a fixed period (δ). These traces are useful to capture the behaviour of discrete systems that execute according to a fixed clock granularity.

In order to address heterogeneity, we need a notion of time refinement:

Definition 2.2 (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 ($\tau' \preceq_\rho \tau$) iff, for every $i \in \mathbb{N}$, $\tau(i) = \tau'(\rho(i))$.

  - We say that τ' *refines* τ ($\tau' \preceq \tau$) 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 ($\lambda' \preceq_\rho \lambda$) iff
  - $\tau' \preceq_\rho \tau$,
  - $\sigma(i) = \sigma'(\rho(i))$ for every $i \in \mathbb{N}$, and
  - $\sigma'(j) = \emptyset$ for every $\rho(i) < j < \rho(i+1)$.

  We also say that λ' *refines* λ ($\lambda' \preceq \lambda$) iff $\lambda' \preceq_\rho \lambda$ for some ρ.

— The *r-closure* of a timed property Λ is $\Lambda^r = \{\lambda' : \exists \lambda \in \Lambda (\lambda' \preceq \lambda)\}$ — the set of all timed traces that refine some timed trace of Λ

— We say that Λ is *closed under time refinement*, or r-closed, iff $\Lambda^r \subseteq \Lambda$ — i.e., Λ contains all the refinements of its timed traces.

We extend the notion of refinement to prefixes considering that, for a prefix π' to refine a prefix π, the two prefixes need to end with same action and time stamp.

We also extend the notion of refinement to timed properties as follows:

— A timed property Λ' refines a timed property Λ ($\Lambda' \preceq \Lambda$) iff, for every $\lambda' \in \Lambda'$, there exists $\lambda \in \Lambda$ such that $\lambda' \preceq \lambda$.

— A timed property Λ' *approximates* a timed property Λ ($\Lambda' \precapprox \Lambda$) iff $\Lambda' \preceq \Lambda$ and, for every $\lambda \in \Lambda$, there exists $\lambda' \in \Lambda'$ such that $\lambda' \preceq \lambda$.

That is, 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 (see Fig. 1).

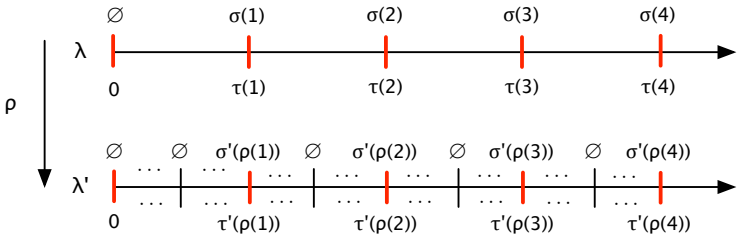

Fig. 1. Time refinement.

A timed property Λ' refines Λ if every trace of Λ' refines some trace of Λ. If, in addition, every trace of Λ has a refinement in Λ', then Λ' approximates Λ.

Proposition 2.1. The following properties of time refinement are useful:

— Given $\tau' \preceq_\rho \tau$ and a timed trace $\lambda = \langle \sigma, \tau \rangle$, there is a unique timed trace $\lambda' = \langle \sigma', \tau' \rangle$ such that $\lambda' \preceq_\rho \lambda$ — we call λ' the refinement of λ over τ'. The trace σ' coincides with σ on λ and executes the empty set of actions everywhere else.

— $\tau_{\delta'} \preceq \tau_\delta$ iff δ is a multiple of δ'.

— Given any two timed properties Λ_1 and Λ_2, if they are r-closed so is their intersection $\Lambda_1 \cap \Lambda_2$.

— If Λ is r-closed, $\Lambda' \preceq \Lambda$ iff $\Lambda' \subseteq \Lambda$.

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 τ_1 and τ_2 being given by the recursion

$$\tau(i+1) = min(\{\tau_1(j) > \tau(i), j \in \mathbb{N}\} \cup \{\tau_2(j) > \tau(i), j \in \mathbb{N}\})$$

together with the base $\tau(0) = 0$. 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 commensurate (have a common divisor), i.e., if 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 (which always exists and can be calculated using Euclid's algorithm provided that δ_1 and δ_2 are commensurate).

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

Definition 2.3 (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 ($A \subseteq B$), then we tend to write $\_|_A$ instead of $\_|_f$.

— 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.

— For every timed property Λ_A over A and every timed property Λ_B over B we write $\Lambda_B \preceq^f \Lambda_A$ (resp. $\Lambda_B \precapprox^f \Lambda_A$) to mean $\Lambda_B \preceq f(\Lambda_A)$ (resp. $\Lambda_B \precapprox f(\Lambda_A)$).

That is, $\_|_f$ projects every trace σ over B to a trace over A by taking the inverse image of the set of actions at every point of σ. In the case of an inclusion $A \subseteq B$, $\_|_A$ forgets the actions of B that are not in A. Notice that the inverse image of a set of actions in B that are not in the range of f is the empty set; if f captures the way in which a machine is part of a network, this means that sets of actions of the network in which the machine is not involved are projected to the machine as the empty set.

A timed property Λ over A is mapped forwards to a timed property over B by taking the set of all traces over B that are projected back to a trace of Λ. Notice that this is different from applying f pointwise to every trace λ of Λ: instead, our construction maps $\lambda = \langle \sigma, \tau \rangle$ to all traces $\langle \sigma', \tau \rangle$ over B such that, for all i, $\sigma(i) = f^{-1}(\sigma'(i))$, which means that every $\sigma'(i)$ may contain any actions of B that are not in the range of f. In particular, we have that $f(\Lambda(A)) = \Lambda(B)$. Again, if f captures the way in which a machine is part of a network, $f(\Lambda)$ will open every trace of Λ to actions of the network in which the machine is not involved.

The following proposition provides a useful property: that projections preserve time refinement.

Proposition 2.2 (Preservation). Let $f{:}A \to B$ be a function (alphabet map).

— Let λ and λ' be two timed traces over B such that $\lambda' \preceq_\rho \lambda$. Then, $\lambda'|_f \preceq_\rho \lambda|_f$.

— Let Λ be a timed property over B. If Λ is r-closed, so is $\Lambda|_f$.

— Let Λ be a timed property over A. If Λ is r-closed, so is $f(\Lambda)$.

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 — because they can bind to other machines at run time, not design time, it would not be realistic to 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 '$p.$'. Note that prefixing defines a bijection between A and its image $p.A$.

2.2. *Timed input/output automata*

In order to model machines, we use timed I/O automata as in (David et al., 2010) except that (1) transitions perform sets of actions instead of single actions and (2) the initial values of clocks are given (in addition to the initial location). Working with sets of actions simplifies the treatment of interconnections through the use of synchronisation sets and gives us for free the empty set as an abstraction of actions performed by the environment that an automaton can observe without being directly involved; transitions performed by the empty set of actions are not required to keep the automaton in the same location because machines can be forced to change state due to clock constraints. Considering that the initial values of clocks are part of the TIOA definition is a generalisation that is useful for modelling the behaviour of a machine from a point in time that is not necessarily that in which the machine started executing.

A timed automaton is defined in terms of a finite set $\mathbb{C}$ of clocks. A *condition* over $\mathbb{C}$ is a finite conjunction of expressions of the form $c \bowtie n$ where $c \in \mathbb{C}$, $\bowtie \in \{\leq, \geq\}$ and $n \in \mathbb{N}$. We denote by $\mathcal{B}(\mathbb{C})$ the set of conditions over $\mathbb{C}$. A *clock valuation* over $\mathbb{C}$ is a mapping $v{:}\ \mathbb{C} \to \mathbb{R}_{\geq 0}$.

Definition 2.4 (TIOA). A timed I/O automaton $\mathcal{A}$ (TIOA) is a tuple

$$\mathcal{A} = \langle Loc, l^\star, \mathbb{C}, v^\star, E, Act, Inv \rangle$$

where:

— Loc is a finite set of locations;
— $l^\star \in Loc$ is the initial location;
— $\mathbb{C}$ is a finite set of clocks;
— $v^\star$ is the initial clock valuation;
— $E \subseteq Loc \times 2^{Act} \times \mathcal{B}(\mathbb{C}) \times 2^{\mathbb{C}} \times Loc$ is a finite set of edges;
— $Act = Act^I \cup Act^O \cup Act^\tau$ is a finite set of actions partitioned into input, output and internal actions, respectively;
— $Inv{:}\ Loc \to \mathcal{B}(\mathbb{C})$ is a mapping that associates an invariant with every location.

In addition, we impose that every TIOA is *open* in the following sense: for all $l \in Loc$,

there is an edge $(l, \emptyset, \phi, \emptyset, l') \in E$ for some tautology ϕ and for some location l' such that $Inv(l')$ is implied by $Inv(l)$.

Given an edge (l, S, C, R, l'), l is the source location, l' is the target location, S is the set of actions executed during the transition, C is the guard (a condition that determines if the transition can be performed), and R is the set of clocks that are reset by the transition. We say that the transition is *enabled* iff C holds.

The requirement that every location is the source of a transition labelled by $\emptyset$ that is always enabled — its guard is a tautology — means that the behaviour of $\mathcal{A}$ is always open to the execution of actions in which it is not involved, i.e., $\mathcal{A}$ does not interfere with the ability of the environment to make progress.

Let v be a *clock valuation* over a set $\mathbb{C}$ of clocks. Given $d \in \mathbb{R}_{\geq 0}$ and a valuation v, we denote by $v+d$ the valuation defined by, for any clock $c \in \mathbb{C}$, $(v+d)(c) = v(c)+d$. Given $R \subseteq \mathbb{C}$ and a clock valuation v, we denote by $v^{\mathbf{R}}$ the valuation where the clocks belonging to R are reset, i.e., such that $v^{\mathbf{R}}(c)=0$ if $c \in R$ and $v^{\mathbf{R}}(c)=v(c)$ otherwise. Given a condition C in $\mathcal{B}(\mathbb{C})$, we use $v \vDash C$ to express that C holds for the clock valuation v.

Definition 2.5 (Execution). Let $\mathcal{A} = \langle Loc, q_0, \mathbb{C}, E, Act, Inv \rangle$ be a TIOA. An *execution* of $\mathcal{A}$ starting in location l_0 and valuation v_0 is an infinite sequence

$$(l_0, v_0, d_0) \xrightarrow{S_0, R_0} (l_1, v_1, d_1) \xrightarrow{S_1, R_1} \dots$$

where, for all i:

(1) $l_i \in Loc$, v_i is a clock valuation over $\mathbb{C}$ and $d_i \in \mathbb{R}_{>0}$;
(2) $S_i \subseteq Act$ and $R_i \subseteq \mathbb{C}$;
(3) for all $0 \leq t \leq d_i$, $v_i + t \vDash Inv(l_i)$;
(4) $v_{i+1} = (v_i + d_i)^{\mathbf{R_i}}$;
(5) there is $(l_i, S_i, C_i, R_i, l_{i+1}) \in E$ such that $v_i + d_i \vDash C_i$.

A *partial execution* is of the form

$$(l_0, v_0, d_0) \xrightarrow{S_0, R_0} \dots \xrightarrow{S_{n-1}, R_{n-1}} (l_n, v_n, d_n)$$

where (1) and (3) hold for all $i \in [0..n]$, and (2), (4) and (5) for all $i \in [0..n-1]$.

That is, each triple (l_i, v_i, d_i) consists of a location l_i, the value v_i of the clocks when l_i is reached at that point of the execution, and the duration d_i for which the automaton remains at l_i before the next transition (which can leave the automaton in the same location, i.e., l_i can be equal to l_{i+1}). During the time that the automaton remains at l_i, the invariant $Inv(l_i)$ holds. Notice that, because every d_i is positive, it is not possible to enter and leave a location instantaneously. The requirement that d_i is positive ensures that different transitions do not occur at the same time; because transitions are labeled with sets of actions, we already have a way of expressing that certain actions occur simultaneously.

A transition out of (l_i, v_i, d_i) happens at the end of d_i units of time and is made by an edge $(l_i, S_i, C_i, R_i, l_{i+1})$ whose guard C_i holds at that time. As a result of the transition, the clocks are updated to $(v_i + d_i)^{\mathbf{R_i}}$ and the location to l_{i+1}. The updated clocks satisfy the invariant of l_{i+1}.

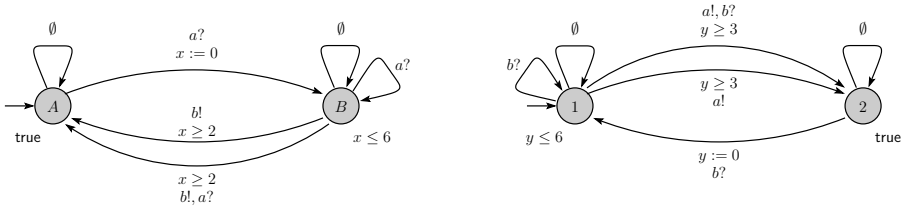

Fig. 2. Two TIOAs: $\mathcal{A}^x$ (left) and $\mathcal{A}^y$(right). Their initial clock valuations are omitted because are implicitly taken to be $\mathbf{0}$.

A pair (l, v) where l is a location and v is a clock valuation is said to be *reachable at time* $T \in \mathbb{R}_{\geq 0}$ if either

(a) $(l, v) = (l^\star, v^\star)$, $T = 0$ and, there exists $d_0{>}0$ such that $t \vDash Inv(l^\star)$ for all $0 \leq t \leq d_0$; or

(b) there exists a partial execution that starts at $(l^\star, v^\star)$ and ends at $(l_n, v_n) = (l, v)$, and $T = \sum_{i \in [0..n-1]} d_i$.

Condition (a), which requires that the initial state be reachable at time 0, is justified by the fact that, according to the definition of execution (Def. 2.5), an automaton has to remain in a state for a positive duration before make a transition to another state. In particular, it needs to be able to remain in the initial state for a positive duration.

Example 2.1. Consider the TIOAs $\mathcal{A}^x$ and $\mathcal{A}^y$ in Fig. 2, where $Act_x^I = \{a\}$, $Act_x^O = \{b\}$, $Act_y^I = \{b\}$, and $Act_y^O = \{a\}$ — for clarity, inputs are decorated with ? and outputs with !. The initial valuation of the clocks is not represented in the figure since it is the default one (every clock starts at 0).

— $\mathcal{A}^x$ waits for receiving a, after which it sends b (possibly receiving a at the same time) within six time units but not before two times units have passed (all a's received in the meanwhile being ignored); then, $\mathcal{A}^x$ waits for receiving a again.

— $\mathcal{A}^y$ leaves the initial state by sending a (possibly receiving b at the same time) within six time units but not before three units have passed; it then waits for receiving b to start again and send a. More b's can be received meanwhile, but they are all ignored.

An example of a partial execution of $\mathcal{A}^x$ is

$$(A, 0, 2) \xrightarrow{\{a\},\{x\}} (B, 0, 3) \xrightarrow{\{b\},\emptyset} (A, 3, 5) \xrightarrow{\{a\},\{x\}} (B, 0, 2)$$

which shows that $(B, 0)$ is reachable at time 2 (after the first transition) and at time 10 $(= 2 + 3 + 5)$ (after three transitions).

We now recall the classical definition of composition of *compatible* TIOAs, which captures partial synchronisation.

Definition 2.6 (Compatibility). Two TIOAs $\mathcal{A}_i{=}\langle Loc_i, l_i^\star, \mathbb{C}_i, v_i^\star, E_i, Act_i, Inv_i\rangle$ are compatible iff $\mathbb{C}_1{\cap}\mathbb{C}_2{=}Act_1^I{\cap}Act_2^I{=}Act_1^O{\cap}Act_2^O{=}Act_1^\tau{\cap}Act_2{=}Act_2^\tau{\cap}Act_1{=}\emptyset$.

Definition 2.7 (Composition). The *composition* of two compatible TIOAs $\mathcal{A}_1$ and $\mathcal{A}_2$

where $\mathcal{A}_i = \langle Loc_i, l_i^\star, \mathbb{C}_i, v_i^\star, E_i, Act_i, Inv_i \rangle$ is

$$\mathcal{A}_1 \parallel \mathcal{A}_2 = \langle Loc, l^\star, \mathbb{C}, v^\star, E, Act, Inv \rangle$$

where:

— $Loc = Loc_1 \times Loc_2$,
— $l^\star = (l_1^\star, l_2^\star)$,
— $\mathbb{C} = \mathbb{C}_1 \cup \mathbb{C}_2$,
— $v^\star = v_1^\star \cup v_2^\star$, i.e., the clock valuation for $\mathbb{C}$ induced by $v_1^\star$ and $v_2^\star$,
— $Act^I = (Act_1^I \backslash Act_2^O) \cup (Act_2^I \backslash Act_1^O)$,
— $Act^O = (Act_1^O \backslash Act_2^I) \cup (Act_2^O \backslash Act_1^I)$,
— $Act^\tau = Act_1^\tau \cup Act_2^\tau \cup (Act_1^I \cap Act_2^O) \cup (Act_1^O \cap Act_2^I)$, and
— for all $(l_1, l_2) \in Loc$:
  – $Inv((l_1, l_2)) = Inv_1(l_1) \wedge Inv_2(l_2)$;
  – $((l_1, l_2), S, C, R, (l_1', l_2')) \in E$ if and only if $C = C_1 \wedge C_2$, $S_i = S \cap Act_i$ $(i = 1, 2)$, $(l_1, S_1, C_1, R_1, l_1') \in E_1$, $(l_2, S_2, C_2, R_2, l_2') \in E_2$, and $R = R_1 \cup R_2$.

Notice that, because the guards of transitions are conjoined, in order for the TIOA that results from the composition to be open (cf. Def. 2.4) we need to require the existence of a transition labeled with $\emptyset$ and a tautological guard, instead of simply a guard with true. Notice also that, by construction, whenever $S \cap Act_1 \neq \emptyset$ and $S \cap Act_2 \neq \emptyset$, all actions on which $\mathcal{A}_1$ and $\mathcal{A}_2$ synchronise (those in $S \cap Act_1 \cap Act_2$) are necessarily inputs on one side and outputs on the other; the composition makes those actions internal. Finally, transitions such that $S \cap Act_i = \emptyset$, which are usually considered as non-synchronising, are in our case handled as synchronising transitions where $\mathcal{A}_i$ performs the empty set of actions (which corresponds to an open semantics).

3. An algebra of timed machines

In order to model systems where applications execute over specific platforms, which implies that they are subject to the clock granularity of the platform, we extend TIOAs to what we call *timed machines*.

3.1. *Timed machines*

A timed machine is a TIOA that executes in the context of a clock granularity δ, i.e., its actions are always executed at time instants that are multiples of δ.

Definition 3.1 (DTIOM). A *discrete timed I/O machine* (DTIOM) is a pair

$$\mathcal{M} = \langle \delta_{\mathcal{M}}, \mathcal{A}_{\mathcal{M}} \rangle$$

where $\delta_{\mathcal{M}} \in \mathbb{R}_{>0}$ and $\mathcal{A}_{\mathcal{M}} = \langle Loc, l^\star, \mathbb{C}, v^\star, E, Act, Inv \rangle$ is a TIOA such that $v^\star$ assigns a multiple of $\delta_{\mathcal{M}}$ to every clock in $\mathbb{C}$.

Definition 3.2 (Execution and behaviour). The *executions* and *partial executions*

of a DTIOM $\mathcal{M} = \langle \delta_{\mathcal{M}}, \mathcal{A}_{\mathcal{M}} \rangle$ are those of $\mathcal{A}_{\mathcal{M}}$ restricted to transitions at every $\delta_{\mathcal{M}}$, i.e.,

$$(l_0, v_0, d_0) \xrightarrow{S_0,R_0} (l_1, v_1, d_1) \xrightarrow{S_1,R_1} \dots$$

such that all the durations d_i are $\delta_{\mathcal{M}}$. Therefore, we represent executions of DTIOMs as sequences

$$(l_0, v_0) \xrightarrow{S_0,R_0} (l_1, v_1) \xrightarrow{S_1,R_1} \dots$$

and call each pair (l_i, v_i) an execution state.

The *behaviour* $[\![\mathcal{M}]\!]$ of $\mathcal{M}$ is the set of executions that start at the initial state $(l^\star, v^\star)$, i.e., those that start in the initial location with each clock c set to $v^\star(c)$.

Notice that a timed machine is not only required to execute its actions at time instants that are multiples of δ but also to make a transition at every multiple of δ. Because TIOAs are open, this does not constitute a limitation: when an automaton reaches a location and, according to the invariant, can remain there for a multiple of δ time units, although it has to make a transition after δ time units, this transition might be the one labelled with $\emptyset$, which is ensured to exist by the openness condition (this condition ensures that there is an edge labelled by the empty set of actions from that location to the same or another location where the automaton can stay for at least δ time units).

Given a machine $\mathcal{M}$ that executes in the context of a clock granularity δ, we require that the machine be able to stay in the initial state during the first δ time units (i.e., $v^\star + t \vDash Inv(l^\star)$ for all $0 \leq t \leq \delta$) in order to consider that the initial state is reachable at time 0. If this is not the case, then the machine has no reachable states. Notice also that, for every reachable state (l, v) in $\mathcal{M}$, v necessarily assigns a multiple of δ to every clock in $\mathbb{C}$.

Proposition and Definition 3.1 (Language). An execution of a DTIOM $\mathcal{M}$

$$(l^\star, v^\star) \xrightarrow{S_0,R_0} (l_1, v_1) \xrightarrow{S_1,R_1} \dots$$

that starts at the initial state defines the $\delta_{\mathcal{M}}$-timed trace $\lambda = \langle \sigma, \tau_{\delta_{\mathcal{M}}} \rangle$ over Act where $\sigma(0) = \emptyset$ and, for all $i \geq 0$, $\sigma(i+1) = S_i$. We denote by $\Lambda_{\mathcal{M}}$ the r-closure of the set of timed traces defined by $[\![\mathcal{M}]\!]$, which we call its *language* and by $\Pi_{\mathcal{M}}$ the set of prefixes of traces in $\Lambda_{\mathcal{M}}$.

The fact that the language of a DTIOM is r-closed means that it contains all possible interleavings of empty observations, thus capturing the behaviour of the DTIOM in any possible environment. This notion of closure can be related to mechanisms, such as stuttering (Abadi and Lamport, 1991), that ensure that components do not constrain their environment.

Example 3.1. Consider the DTIOM $\mathcal{M}^x = \langle \delta_x, \mathcal{A}^x \rangle$ with $\delta_x = 2$ and $\mathcal{A}^x$ as in Ex. 2.1. The partial execution of $\mathcal{A}^x$ given in Ex. 2.1 is not a partial execution of $\mathcal{M}^x$ as it does not respect the granularity $\delta_x = 2$. An example of a partial execution of $\mathcal{M}^x$ is

$$(A, 0) \xrightarrow{\{a\},\{x\}} (B, 0) \xrightarrow{\emptyset,\emptyset} (B, 2) \xrightarrow{\{b\},\emptyset} (A, 4)$$

Note that this means that a was executed at time 2 — $\mathcal{M}^x$ remaining in the initial state for $\delta_x = 2$ units of time, nothing was executed at time 4, and b was executed at time 6.

3.2. *Composition and refinement of timed machines*

Having in mind the ability to model systems that can be interconnected at run time, not design time, we are interested in a notion of composition of DTIOMs that allows us to compose two machines $\mathcal{M}_1$ and $\mathcal{M}_2$ that may have started executing before the time they are composed, the result being a machine that models their joint behaviour henceforth. This means that, when the machines are composed, each is at a reachable state that is not necessarily its initial state. Therefore, we define a composition operator that is parametric on a pair of *composition conditions*, identifying the state (l_i, v_i) in which each machine is, and the time t_i at which this state was reached.

We start by addressing the simplest case in which the two machines have the same clock granularity. This notion of composition of DTIOMs can be defined in terms of the composition of the corresponding DTIOAs as follows:

Definition 3.3 (Composition). Given two DTIOMs $\mathcal{M}_i = \langle \delta, \mathcal{A}_i \rangle$ – $i = 1, 2$ – such that $\mathcal{A}_1$ and $\mathcal{A}_2$ are compatible, and triples (l_i, v_i, t_i) such that (l_i, v_i) is a state of $\mathcal{M}_i$ reachable at time t_i, we define

$$\prod_{i=1,2}^{(l_i,v_i,t_i)} \mathcal{M}_i$$

as the DTIOM $\langle \delta, \mathcal{A} \rangle$ where $\mathcal{A}$ is the TIOA we obtain by replacing the initial location and clock valuation of $\mathcal{A}_1 \parallel \mathcal{A}_2$ with (l_1, l_2) and $v_1 \cup v_2$, respectively.

The composition of two machines that execute in the context of the same clock granularity results in a new machine that models their joint behaviour henceforth: the resulting machine starts its execution at the locations and with the clock valuations of the individual machines at composition time.

As remarked before, the fact that each (l_i, v_i) is a reachable state of $\mathcal{M}_i$ ensures that both v_1 and v_2 assign a multiple of δ to every clock and, hence, $\langle \delta, \mathcal{A} \rangle$ is indeed a DTIOM.

Considering now the situation of two DTIOMs $\langle \delta_1, \mathcal{A}_1 \rangle$ and $\langle \delta_2, \mathcal{A}_2 \rangle$ that have different clock granularities, if δ_1 and δ_2 admit a common multiple, i.e., $\delta_1 \cdot n = \delta_2 \cdot m$ for given $n, m \in \mathbb{N}_{>0}$, then they will be able to synchronise from time to time (more precisely, at the common multiples). We investigate now how a composition operator can be defined that captures that sort of synchronisation.

The idea is to refine both timed machines to a common clock granularity and then compose the refinements as in Def. 3.3. Intuitively, given a timed machine $\mathcal{M} = \langle \delta, \mathcal{A} \rangle$, we define its *k-refinement* $\mathcal{M}_k = \langle \delta/k, \mathcal{A}_k \rangle$ by dividing every location of $\mathcal{A}$ in k copies such that the original transitions are performed in the last 'tick', all previous 'ticks' performing no actions and, therefore, being open for synchronisation with a machine that ticks with a granularity δ/k.

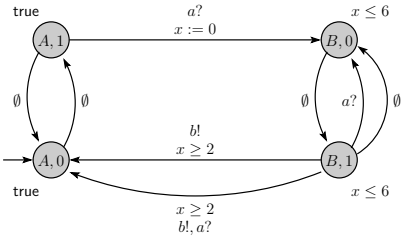

Fig. 3. The refinement $\mathcal{A}_2^x$ of $\mathcal{A}^x$

Definition 3.4 (Refinement). Given a TIOA $\mathcal{A} = \langle Loc, l^\star, \mathbb{C}, v^\star, E, Act, Inv\rangle$ and $k \in \mathbb{N}_{>0}$, its *k-refinement* is the TIOA $\mathcal{A}_k = \langle Loc_k, {l^\star}_k, \mathbb{C}, v^\star, E_k, Act, Inv_k\rangle$ where:

— $Loc_k = Loc \times [0..k-1]$;
— ${l^\star}_k = (l^\star, 0)$;
— $Inv_k(l,i) = Inv(l)$;
— for every (l, S, C, R, l') of E, E_k comprises of the edge $((l, k-1), S, C, R, (l', 0))$ and all edges of the form $((l,i), \emptyset, \mathsf{true}, \emptyset, (l, i+1))$, $i \in [0..k-2]$.

Given a timed machine $\mathcal{M} = \langle \delta, \mathcal{A}\rangle$, its *k-refinement* is $\mathcal{M}_k = \langle \delta/k, \mathcal{A}_k\rangle$.

Notice that $\mathcal{A}_k$ is indeed a TIOA: it contains all the required empty transitions, and the initial clock valuation of $\mathcal{A}_k$, being that of $\mathcal{A}$, assigns multiples of δ to every clock (which are necessarily multiples of δ/k).

Machines and their refinements are related through what we call an *approximation*:

Proposition and Definition 3.2 (Approximation). Given a timed machine $\mathcal{M} = \langle \delta, \mathcal{A}\rangle$ and $k \in \mathbb{N}_{>0}$:

— Every execution of $\mathcal{M}$ defines a unique execution of $\mathcal{M}_k$, which we call its refinement.
— Because $\Lambda_\mathcal{M}$ is r-closed, $\Lambda_{\mathcal{M}_k} \subseteq \Lambda_\mathcal{M}$ and, hence, $\Lambda_{\mathcal{M}_k} \preceq \Lambda_\mathcal{M}$.
— For every $\lambda \in \Lambda_\mathcal{M}$ there is $\lambda' \in \Lambda_{\mathcal{M}_k}$ such that $\lambda' \preceq \lambda$ and, hence, $\Lambda_{\mathcal{M}_k} \precapprox \Lambda_\mathcal{M}$. We say that $\mathcal{M}_k$ approximates $\mathcal{M}$ and write $\mathcal{M}_k \precapprox \mathcal{M}$.

More generally, for arbitrary DTIOM $\mathcal{M}$ and $\mathcal{M}'$ that have a common alphabet (i.e., $Act_{\mathcal{M}'} = Act_\mathcal{M}$), we define $\mathcal{M}' \preceq \mathcal{M}$ to mean that $\delta_\mathcal{M}$ is a multiple of $\delta_{\mathcal{M}'}$ and $\Lambda_{\mathcal{M}'} \preceq \Lambda_\mathcal{M}$; and we define $\mathcal{M}' \precapprox \mathcal{M}$ to mean that $\delta_\mathcal{M}$ is a multiple of $\delta_{\mathcal{M}'}$ and $\Lambda_{\mathcal{M}'} \precapprox \Lambda_\mathcal{M}$. We extend this to the case in which there is an injection $\xi : Act_\mathcal{M} \to Act_{\mathcal{M}'}$ and define $\mathcal{M}' \precapprox^\xi \mathcal{M}$ as in Def. 2.3.

That is, the language of $\mathcal{M}_k$ refines and is an approximation of the language of $\mathcal{M}$, which we take to mean that $\mathcal{M}_k$ refines and is an approximation of $\mathcal{M}$.

Example 3.2. Consider the TIOA $\mathcal{A}^x$ in Fig. 2 and the corresponding DTIOM $\mathcal{M}^x$ defined in Ex. 3.1, which has granularity 2. Its refinement to a DTIOM with granularity 1 is $\mathcal{M}_2^x = \langle 1, \mathcal{A}_2^x\rangle$, with $\mathcal{A}_2^x$ given in Fig. 3. The refinement of the partial execution of $\mathcal{M}^x$ given in Ex. 3.1 is:

$$((A,0),0) \xrightarrow{\emptyset,\emptyset} ((A,1),1) \xrightarrow{\{a\},\{x\}} ((B,0),0) \xrightarrow{\emptyset,\emptyset} ((B,1),1) \xrightarrow{\emptyset,\emptyset} ((B,0),2) \xrightarrow{\emptyset,\emptyset} ((B,1),3) \xrightarrow{\{b\},\emptyset} ((A,0),4)$$

We can now extend the composition of two timed machines to the case where their clock granularities are commensurable (have a common divisor):

Definition 3.5 (Heterogeneous compatibility). Two DTIOMs $\mathcal{M}_i = \langle \delta_i, \mathcal{A}_i \rangle$ – $i = 1, 2$ – are said to be δ-*compatible* (where $\delta \in \mathbb{R}_{>0}$) if (a) $\mathcal{A}_1$ and $\mathcal{A}_2$ are compatible, and (b) δ is a common divisor of δ_1 and δ_2. They are said to be *compatible* if they are δ-*compatible* for some δ, i.e., if $\mathcal{A}_1$ and $\mathcal{A}_2$ are compatible and δ_1 and δ_2 are commensurate.

Definition 3.6 (Heterogeneous composition). The δ-*composition* of two δ-*compatible* DTIOMs $\mathcal{M}_1$ and $\mathcal{M}_2$ at (l_i, v_i, t_i) – $i = 1, 2$ – such that each (l_i, v_i) is a state of $\mathcal{M}_i$ reachable at time t_i is defined as follows:

$$\Big\|_{\delta}^{(l_i,v_i,t_i)}_{i=1,2} \mathcal{M}_i \triangleq \Big\|_{i=1,2}^{((l_i,0),v_i,t_i)} \mathcal{M}_{i(\delta_i/\delta)}$$

If δ is the greatest common divisor of δ_1 and δ_2, we use the notation $\|_{i=1,2}^{(l_i,v_i,t_i)} \mathcal{M}_i$ and simply refer to *the composition* of $\mathcal{M}_1$ and $\mathcal{M}_2$ at (l_i, v_i).

Notice that if $\mathcal{A}_1$ and $\mathcal{A}_2$ are compatible, so are $\mathcal{A}_{1(\delta_1/\delta)}$ and $\mathcal{A}_{2(\delta_2/\delta)}$. The executions of $\mathcal{M}_{i(\delta_i/\delta)}$ that refine each execution of $\mathcal{M}_i$ ensure that the fact that (l_i, v_i) is reachable at time t_i in $\mathcal{M}_i$ implies that $((l_i, 0), v_i)$ is reachable at time t_i in $\mathcal{M}_{i(\delta_i/\delta)}$.

Example 3.3. Consider $\mathcal{M}^x$ as defined in Ex. 3.1, i.e., $\mathcal{M}^x = \langle \delta_x, \mathcal{A}^x \rangle$ with $\delta_x = 2$, and $\mathcal{M}^y = \langle \delta_y, \mathcal{A}^y \rangle$ with $\delta_y = 1$ and $\mathcal{A}^y$ as in Ex. 2.1. Consider also the initial states $(A, 0)$ and $(1, 0)$ of these machines, which are reachable at time 0. Because $\mathcal{A}^x$ and $\mathcal{A}^y$ are compatible and δ_x and δ_y have a common divisor ($\delta = 1$), we can compute their composition at $(A, 0, 0)$ and $(1, 0, 0)$. The first step consists in refining $\mathcal{A}^x$ into $\mathcal{A}^x_2$ (Fig. 3). We have that

$$\mathcal{M}^x \Big\|^{(A,0,0),(1,0,0)} \mathcal{M}^y = \langle 1, \mathcal{A}^{x,y} \rangle$$

where $\mathcal{A}^{x,y} = \mathcal{A}^x_2 \| \mathcal{A}^y$ is given in Fig. 4. Notice that actions a and b are synchronised and, hence, made internal in the composition, which we denote by a^τ and b^τ, respectively. Notice also that, since the composition is made at the time the machines start their execution, the replacement of the initial location and clock valuation of $\mathcal{A}^x_2 \| \mathcal{A}^y$ by those defined by the composition conditions do not change the TIOA.

Consider now the composition of $\mathcal{M}^x$ and $\mathcal{M}^y$ with $\mathcal{M}^x$ in the state $(A, 4)$ reached at time 6 (for example, after the partial execution presented in Ex. 3.1) and $\mathcal{M}^y$ in its initial state, reached at time 0, as before. This composition is

$$\mathcal{M}^x \Big\|^{(A,4,6),(1,0,0)} \mathcal{M}^y = \langle 1, \mathcal{A}'^{x,y} \rangle$$

where $\mathcal{A}'^{x,y}$ is the TIOA that results from replacing the initial clock valuation of $\mathcal{A}^{x,y}$ with $\{x \mapsto 4, y \mapsto 0\}$.

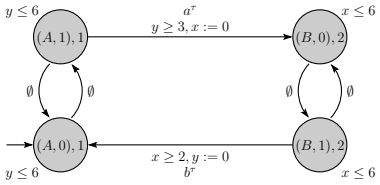

Fig. 4. The TIOA $\mathcal{A}^{x,y}$ of $\mathcal{M}^x \,||^{(A,0,0),(1,0,0)}\, \mathcal{M}^y$

3.3. *Büchi representation of timed machines*

In order to check structural properties of DTIOMs, namely properties formulated in terms of reachable states, it is useful to be able to construct Büchi-automata "equivalents" of DTIOMs.

Let $\mathcal{A} = \langle Loc, l^\star, \mathbb{C}, v^\star, E, Act, Inv\rangle$ be a TIOA. Given a clock $c \in \mathbb{C}$, we use $\mathsf{Max}^{\mathcal{A}}(c)$ to denote the maximal constant to which c is compared in the guards and invariants of $\mathcal{A}$. Formally,

$$\mathsf{Max}^{\mathcal{A}}(c) = \max\{n \in \mathbb{N} \mid (c \bowtie n) \in Inv(l) \text{ for some } l \in Loc, \text{or}$$

$$(c \bowtie n) \in C \text{ for some } (l, S, C, R, l') \in E\}.$$

Let $\mathcal{M} = \langle \delta, \mathcal{A}\rangle$ be a DTIOM. We first propose an equivalence class on clock valuations, which we define as follows: $v \sim v'$ iff, for all $c \in \mathbb{C}$, either $v(c) = v'(c)$ or $v(c) > \mathsf{Max}^{\mathcal{A}}(c)$ and $v'(c) > \mathsf{Max}^{\mathcal{A}}(c)$.

Lemma 3.1. For all locations l and clock valuations v and v' such that $v \sim v'$, the states (l, v) and (l, v') of $\mathcal{M}$ have equivalent outgoing transitions, i.e., whenever state (l, v) can take a transition $(l, S, C, R, l') \in E$ and end up in (l', v''), state (l, v') can take the same transition and end up in (l', v''') with $v'' \sim v'''$, and vice versa.

Definition 3.7 (Büchi equivalent). The Büchi-equivalent of a DTIOM $\mathcal{M} = \langle \delta, \mathcal{A}\rangle$ where $\mathcal{A} = \langle Loc, l^\star, \mathbb{C}, v^\star, E, Act, Inv\rangle$, is the Büchi automaton $\mathcal{B}_{\mathcal{M}} = \langle Q, q_0, 2^{Act}, \to, Q\rangle$ defined as follows:

— The state-space of $\mathcal{B}_{\mathcal{M}}$ is $Q = Loc \times \prod_{c\in\mathbb{C}}[0 \,..\, \lfloor\frac{\mathsf{Max}^{\mathcal{A}}(c)}{\delta}\rfloor + 1]$. A state in $\mathcal{B}_{\mathcal{M}}$ is thus a pair $(l, \boldsymbol{\nu})$ where l is a location and $\boldsymbol{\nu}$ a clock valuation that takes its values in the interval $[0 \,..\, \lfloor\frac{\mathsf{Max}^{\mathcal{A}}(c)}{\delta}\rfloor + 1]$. The idea is to build clock regions of size δ that represent the number of δ-"ticks" elapsed since the last reset for each clock — $\boldsymbol{\nu}$ identifies such a region.

— The initial state of $\mathcal{B}_{\mathcal{M}}$ is $q_0 = (l^\star, \boldsymbol{\nu})$ where $\boldsymbol{\nu}(c)=\frac{v^\star(c)}{\delta}$ if $v^\star(c) \le \mathsf{Max}^{\mathcal{A}}(c)$ and $\boldsymbol{\nu}(c)=\lfloor\frac{\mathsf{Max}^{\mathcal{A}}(c)}{\delta}\rfloor + 1$ otherwise; i.e., it is the initial location of $\mathcal{M}$ associated with the clock region defined by the initial clock valuation of $\mathcal{M}$.

— Because we are interested in all infinite words, all states of $\mathcal{B}_{\mathcal{M}}$ are made accepting.

— Finally, $(l, \boldsymbol{\nu}) \xrightarrow{S} (l', \boldsymbol{\nu}')$ iff there exists a transition $(l, S, C, R, l') \in E$ such that

(i) for all $0 \le t \le \delta$, $\boldsymbol{\nu} \cdot \delta + t \models Inv(l)$,

(ii) $\boldsymbol{\nu} \cdot \delta + \delta \models C$,

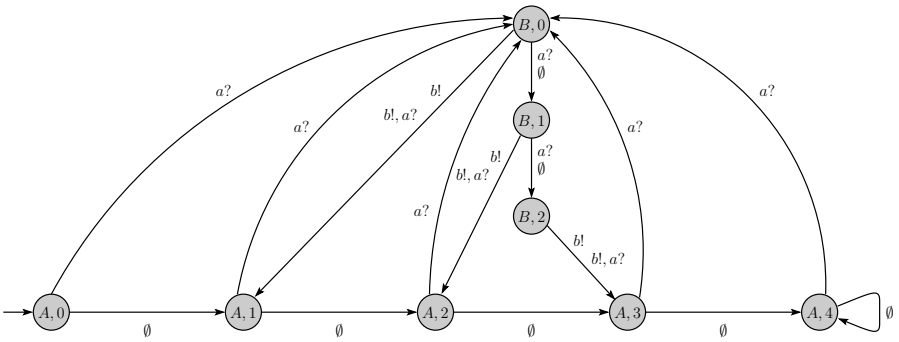

Fig. 5. The Büchi automaton $\mathcal{B}_{\mathcal{M}^x}$ corresponding to the DTIOM $\mathcal{M}^x$ from Ex. 3.1 (unreachable states have been removed)

(iii) for all $c \in \mathbb{C}$, $\boldsymbol{\nu}'(c) = \left\{ \begin{array}{ll} 0 & \text{if } c \in R \\ \boldsymbol{\nu}(c) & \text{if } c \notin R \text{ and } \boldsymbol{\nu}(c) = \lfloor \frac{\mathsf{Max}^{\mathcal{A}}(c)}{\delta} \rfloor + 1 \\ \boldsymbol{\nu}(c) + 1 & \text{otherwise} \end{array} \right\}$

(iv) $\boldsymbol{\nu}' \cdot \delta \models Inv(l')$.

The size of $\mathcal{B}_{\mathcal{M}}$, which we denote by $|\mathcal{B}_{\mathcal{M}}|$, is in $O(|Loc| \cdot (\lfloor \frac{\mathsf{Max}}{\delta} \rfloor + 2)^{|\mathbb{C}|})$, where $|Loc|$ and $|\mathbb{C}|$ are the size of Loc and the number of clocks, respectively, and $\mathsf{Max} = \max\{\mathsf{Max}^{\mathcal{A}}(c) \mid c \in \mathbb{C}\}$ is the maximal constant considered in all constraints and invariants of $\mathcal{A}$.

The Büchi automaton $\mathcal{B}_{\mathcal{M}}$ is equivalent to $\mathcal{M}$ in the following sense:

Theorem 3.1. For all action sequences σ over Act, $\langle \sigma, \tau_{\delta_{\mathcal{M}}} \rangle$ is a timed trace defined by an execution in $[\![\mathcal{M}]\!]$ iff the infinite sequence $\sigma(1)\sigma(2)\ldots$ is in the language of $\mathcal{B}_{\mathcal{M}}$.

Example 3.4. Consider the DTIOM $\mathcal{M}^x$ from Ex. 3.1. The maximal constant to which clock x is compared is $\mathsf{Max}^{\mathcal{A}^x}(x) = 6$. Since $\delta_x = 2$, the corresponding Büchi automaton $\mathcal{B}_{\mathcal{M}^x}$ has 10 states (its state space is $\{A, B\} \times \{0, \ldots, 4\}$). $\mathcal{B}_{\mathcal{M}^x}$ is given in Fig. 5. Unreachable states have been removed.

4. Consistency and feasibility of timed machines

In this section, we investigate two important properties of DTIOMs as models of systems: consistency (in the sense that they generate a non-empty language) and feasibility (in the sense that they generate a non-empty language no matter what inputs they receive). We are especially interested in conditions under which consistency/feasibility are preserved by composition. This is because, in order to model run-time interconnections of systems, one should be able to guarantee that a composition of DTIOMs is consistent/feasible *without* having to actually calculate their composition and analyse it.

4.1. *Consistency*

Definition 4.1 (Consistency). A DTIOM $\mathcal{M}$ is said to be *consistent* if $\Lambda_{\mathcal{M}} \neq \emptyset$.

Notice that consistency is preserved by refinement:

Proposition 4.1. Let $k \in \mathbb{N}_{>0}$. A DTIOM $\mathcal{M}$ is consistent iff its k-refinement $\mathcal{M}_k$ is consistent. More generally, for arbitrary DTIOM $\mathcal{M}$ and $\mathcal{M}'$,

— if $\mathcal{M}' \preceq \mathcal{M}$ and $\mathcal{M}'$ is consistent, then so is $\mathcal{M}$, and
— if $\mathcal{M}' \precapprox \mathcal{M}$, then $\mathcal{M}'$ is consistent iff $\mathcal{M}$ is consistent.

We now investigate sufficient conditions for a DTIOM to be consistent.

Definition 4.2 (Initializable). A DTIOM $\mathcal{M}$ is said to *be initializable* if, for every $0 \leq t \leq \delta_{\mathcal{M}}$, $(l^\star, v^\star + t) \vDash Inv(l^\star)$.

That is, a DTIOM is initializable if it can stay in the initial state until the first tick of the clock.

Another important property is that a DTIOM can make independent progress (which we adapt from (David et al., 2010)) in the sense that it is able to make a transition from any reachable state without forcing the environment to provide any input:

Definition 4.3 (Independent progress). A DTIOM $\mathcal{M}$ is said to *make independent progress* if, for every reachable state (l, v), there is an edge (l, A, C, R, l') such that:

(a) $A \subseteq Act^O_{\mathcal{M}} \cup Act^\tau_{\mathcal{M}}$
(b) $v + \delta_{\mathcal{M}} \vDash C$
(c) for all $0 \leq t \leq \delta_{\mathcal{M}}$, $(v + \delta_{\mathcal{M}})^{\mathbf{R}} + t \vDash Inv(l')$

As an example, both $\mathcal{M}^x$ and $\mathcal{M}^y$ given in Ex. 3.1 are initializable and make independent progress.

Proposition 4.2. Any initializable DTIOM that makes independent progress is consistent.

Notice that checking that a timed machine makes independent progress requires only the analysis of properties of its reachable states. In practice, this can be done using a syntactic check on the Büchi automaton as constructed in Sec. 3.3: a given DTIOM $\mathcal{M}$ makes independent progress iff all reachable states $(l, \boldsymbol{\nu})$ of the equivalent Büchi automaton $\mathcal{B}_{\mathcal{M}}$ have at least one outgoing transition $(l, \boldsymbol{\nu}) \xrightarrow{A} (l', \boldsymbol{\nu}')$ with $A \subseteq Act^O_{\mathcal{M}} \cup Act^\tau_{\mathcal{M}}$. Because $\mathcal{B}_{\mathcal{M}}$ has only finitely many states, denoted by $|\mathcal{B}_{\mathcal{M}}|$, and finitely many transitions, denoted by $|E_{\mathcal{M}}|$, making independent progress can be checked in time $O(|\mathcal{B}_{\mathcal{M}}| \cdot |E_{\mathcal{M}}|)$.

4.2. *Compositional consistency checking*

In order to investigate conditions that can guarantee compositionality of consistency checking, we start by remarking that the fact that two DTIOMs $\mathcal{M}_1$ and $\mathcal{M}_2$ are such that δ_1 and δ_2 are commensurate simply means that we can find a clock granularity in which we can accommodate the transitions that the two DTIOMs perform: by itself, this does not ensure that the two DTIOMs can jointly execute their input/output synchronisation pairs. For example, if $\delta_1 = 2$ and $\delta_2 = 3$ and $\mathcal{M}_2$ only performs non-empty actions at odd multiples of 3, the two machines will not be able to agree on their input/output

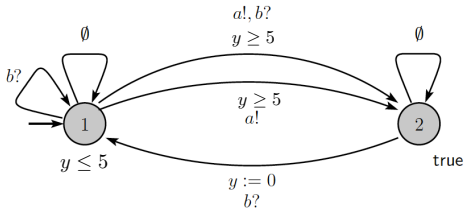

Fig. 6. The TIOA $\mathcal{A}'$.

synchronisation pairs. For the DTIOMs to actually be able to interact with each other it is necessary that their input/output synchronisation pairs can be performed on a common multiple of δ_1 and δ_2.

Definition 4.4 (Cooperative). A DTIOM $\mathcal{M}$ is said to be *cooperative in relation to* $Q \subseteq Act_{\mathcal{M}}$ and a multiple δ of $\delta_{\mathcal{M}}$ if the following holds for every (l, v) reachable at a time T such that $(T + \delta_{\mathcal{M}})$ is not a multiple of δ:

> for every edge $(l, A, C, R, l') \in E_{\mathcal{M}}$ such that $v+\delta_{\mathcal{M}} \vDash C$ and $(v+\delta_{\mathcal{M}})^{\mathbf{R}}+t \vDash Inv_{\mathcal{M}}(l')$ for all $0 \leq t \leq \delta_{\mathcal{M}}$ — i.e., the machine makes a transition at a time that is not a multiple of δ — there exists an edge $(l, A \backslash Q, C', R', l'')$ such that $v + \delta_{\mathcal{M}} \vDash C'$ and, for all $0 \leq t \leq \delta_{\mathcal{M}}$, $(v + \delta_{\mathcal{M}})^{\mathbf{R}'} + t \ \vDash \ Inv_{\mathcal{M}}(l'')$ — i.e., the machine can make an alternative transition that executes all the actions of the original transition except those in Q.

Essentially, being cooperative in relation to Q and δ means that the machine will not force transitions that perform actions in Q at times that are not multiples of δ. In practice, this can be verified using a syntactic check on the states of the equivalent Büchi automaton that can be reached with a number of transitions n such that $n + 1$ is not a multiple of $\delta/\delta_{\mathcal{M}}$. This check can be done in time $O(\frac{\delta}{\delta_{\mathcal{M}}} \cdot |\mathcal{B}_{\mathcal{M}}| \cdot |E_{\mathcal{M}}|^2)$, where $\mathcal{B}_{\mathcal{M}}$ is the Büchi automaton defined in Sec. 3.3.

Example 4.1. $\mathcal{M}^y$ from Ex. 3.1 is cooperative in relation to $\{a, b\}$ and $\delta = 2$. In contrast, the machine $\mathcal{M}'$ with $\delta' = 1$ and the TIOA $\mathcal{A}'$ presented in Fig. 6 is not cooperative in relation to $\{a, b\}$ and $\delta = 2$. Indeed, the fact that the state corresponding to the location 1 is reached at time 4 enables the transition $(1, \{a\}, y \geq 5, \emptyset, 2)$, which cannot be replaced by $(1, \emptyset, \mathsf{true}, \emptyset, 1)$ because the last condition — for all $0 \leq t \leq 1 = \delta_y$, $5 + t \leq 5$ — is violated. Because the machine $\mathcal{M}'$ forces the output of a at time 5, it is easy to conclude that its composition with the machine $\mathcal{M}^x$ from Ex. 3.1 (which has a clock granularity $\delta_x = 2$) at the time the machines start their execution results in an inconsistent DTIOM.

In relation to the composition of two DTIOMs $\mathcal{M}_1$ and $\mathcal{M}_2$, the idea is to require that a common multiple of δ_1 and δ_2 exists such that both DTIOMs are cooperative in relation to $Act_{\mathcal{M}_1} \cap Act_{\mathcal{M}_2}$. However, this is not enough to guarantee that the two DTIOMs can actually work together: we need to ensure that if, say, $\mathcal{M}_1$ wants to output an action, $\mathcal{M}_2$ can accept it.

Definition 4.5 (DP-enabled). A DTIOM $\mathcal{M}$ is said to be *DP-enabled in relation to* $J \subseteq Act^I_{\mathcal{M}}$ and δ multiple of $\delta_{\mathcal{M}}$ if the following property holds for every $B \subseteq J$ and state (l, v) reachable at a time T such that $(T + \delta_{\mathcal{M}})$ is a multiple of δ:

> for every edge $(l, A, C, R, l') \in E_{\mathcal{M}}$ such that $v + \delta_{\mathcal{M}} \vDash C$ and, for all $0 \leq t \leq \delta_{\mathcal{M}}$, $(v + \delta_{\mathcal{M}})^{\mathbf{R}} + t \vDash Inv_{\mathcal{M}}(l')$ — i.e., the machine can make a transition — there exists an edge $(l, B \cup (A \backslash J), C', R', l'')$ such that $v + \delta_{\mathcal{M}} \vDash C'$ and, for all $0 \leq t \leq \delta_{\mathcal{M}}$, $(v + \delta_{\mathcal{M}})^{\mathbf{R}'} + t \vDash Inv_{\mathcal{M}}(l'')$ — i.e., the machine can make an alternative transition that accepts instead B as inputs and still performs the same outputs (and inputs outside J).

That is, a DTIOM is DP-enabled in relation to a set of inputs J and a multiple δ of its clock granularity if, whenever it leaves a reachable state at a multiple of δ, it can do so by accepting any subset of J, and if its outputs are independent of the inputs in J that it receives. Both $\mathcal{M}^x$ and $\mathcal{M}^y$ from Ex. 3.1 are DP-enabled in relation to the set of input actions ($\{a\}$ and $\{b\}$, respectively) and $\delta_x = 2$.

Notice that being DP-enabled is not the same as being input-enabled (Kaynar et al., 2006) in that we work with sets of actions (synchronisation sets), not just individual actions as labels of edges. Because, in our case, inputs and outputs can occur together, we need to ensure that there is no dependency between those that are included in the same synchronisation set. This notion was introduced in (Delahaye et al., 2013) where we used a communication model based on delivery and publication of messages, hence the acronym *DP*.

Being DP-enabled can be verified using a syntactic check on states of the equivalent Büchi automaton that can be reached in a number of steps n such that $n+1$ is a multiple of $\delta/\delta_{\mathcal{M}}$. This can be done in $O(\frac{\delta}{\delta_{\mathcal{M}}} \cdot |\mathcal{B}_{\mathcal{M}}| \cdot |E_{\mathcal{M}}|^2 \cdot 2^{|Act^I_{\mathcal{M}}|})$, with $|\mathcal{B}_{\mathcal{M}}|$ as given in Sec. 3.3.

We now investigate how the composition of two DTIOMs can be shown to be consistent. We start by analysing how properties behave under refinement and composition.

Lemma 4.1. If a DTIOM $\mathcal{M}$ is initializable (makes independent progress, is DP-enabled / cooperative in relation to J and δ'), then so is $\mathcal{M}_k$ for all $k \in \mathbb{N}_{>0}$.

That is, refinement preserves initializability, independent progress, being DP-enabled and being cooperative.

Lemma 4.2. Let $\mathcal{M}_i = \langle \delta_i, \mathcal{A}_i \rangle$, for $i = 1, 2$, be two δ-compatible DTIOMs. Let $\mathcal{M} = {}_{\delta}\Big|\Big|^{(l_i, v_i, t_i)}_{i=1,2} \mathcal{M}_i$ where, for $i = 1, 2$, (l_i, v_i) is a state of $\mathcal{M}_i$ reachable at time t_i. Let δ'_1 be a multiple of δ_1 such that t_1 is a multiple of δ'_1.

(a) $\mathcal{M}$ is initializable.
(b) If $\mathcal{M}_1$ is DP-enabled in relation to $J \subseteq Act^I_1$ and δ'_1, then $\mathcal{M}$ is DP-enabled in relation to $J \setminus Act^O_2$ and δ'_1.
(c) If $\mathcal{M}_1$ is cooperative in relation to $Q \subseteq Act^O_1 \setminus Act^I_2$ and δ'_1, then $\mathcal{M}$ is cooperative in relation to Q and δ'_1.

Notice that for the preservation of being DP-enabled we need to remove from J all actions that were used for synchronising with $\mathcal{M}_2$, which are necessarily in Act^O_2. This is

because they become internal to the composition and, therefore, are no longer available for synchronisation. The preservation of being cooperative is relative to a set of actions that are not used for synchronisation. Both results, stated for a specific δ_1' multiple of δ_1, only hold if $\mathcal{M}_1$ is in a state reached at a time multiple of δ_1'.

Theorem 4.1 (Compositionality). Let $\mathcal{M}_1$ and $\mathcal{M}_2$ be δ-compatible DTIOMs that can make independent progress. Let $\mathcal{M} = {}_{\delta}\|\|_{i=1,2}^{(l_i,v_i,t_i)}\, \mathcal{M}_i$ where, for $i = 1, 2$, (l_i, v_i) is a state of $\mathcal{M}_i$ reachable at time t_i.

If, for some δ' multiple of δ_1 and δ_2 such that t_1 and t_2 are multiples of δ', $\mathcal{M}_1$ is DP-enabled in relation to $Act_1^I \cap Act_2^O$ and δ', $\mathcal{M}_2$ is DP-enabled in relation to $Act_2^I \cap Act_1^O$ and δ', and both $\mathcal{M}_1$ and $\mathcal{M}_2$ are δ'-cooperative in relation to $Act_1 \cap Act_2$, then $\mathcal{M}$ is initializable and makes independent progress (and, hence, by Prop. 4.2, is consistent).

This result allows us to conclude that the machines $\mathcal{M}^x$ and $\mathcal{M}^y$ presented in Ex. 3.1 can work together if, at composition time, both machines are in states reached at a time multiple of 2. This is because, as noted before, $\mathcal{M}^x$ and $\mathcal{M}^y$ are DP-enabled in relation to $\delta' = 2$ and $\{a\}$ and $\{b\}$, respectively, and are cooperative in relation to $\{a, b\}$ and $\delta' = 2$. In particular, this means that both compositions we considered in Ex. 3.3 are both consistent.

The results stated in this section can also be used to ensure, in a compositional way, that the composition of three or more machines is consistent. Let

$$\mathcal{M} = {}_{\delta}\|\|_{i=1,2}^{(l_i,v_i,t_i)}\, \mathcal{M}_i$$

and let δ_i' be a multiple of δ_i such that t_i is a multiple of δ_i'. From Lem. 4.2, if $\mathcal{M}_i$ is DP-enabled in relation to $J \subseteq Act_i^I$ and δ_i', $\mathcal{M}$ is DP-enabled in relation to $J \backslash Act_{\overline{i}}^O$ and δ_i', with $\overline{1}$=2 and $\overline{2}$=1. Moreover, if $\mathcal{M}_i$ is cooperative in relation to $Q \subseteq Act_i^O \setminus Act_{\overline{i}}^I$ and δ_i', $\mathcal{M}$ is cooperative in relation to Q and δ_i'. This implies that, in order to ensure that the composition of $\mathcal{M}$ with a third machine $\mathcal{M}_3$ (which can itself be the result of a composition) is consistent, we can verify the required properties (being DP-enabled and cooperative) over the component machines: we do not need to make checks over the machines resulting from the compositions, i.e., our method is compositional.

4.3. *Feasibility*

The property of being DP-enabled is related to a stronger notion of consistency called 'feasibility': whereas consistency guarantees the existence of an execution, feasibility requires that, no matter what inputs the machine receives from its environment, it can produce an execution.

Definition 4.6 (Feasible). A DTIOM $\mathcal{M}$ is said to be *feasible in relation to* $J \subseteq Act_{\mathcal{M}}^I$ and a multiple δ of $\delta_{\mathcal{M}}$ if it is initializable and, for every δ-timed trace λ over J and state (l, v) reachable at a time T such that $(T + \delta_{\mathcal{M}})$ is a multiple of δ, there is an execution starting at (l, v) that generates a $\delta_{\mathcal{M}}$-timed trace λ' such that $\lambda'|_J \preceq \lambda$, where $\lambda'|_J$ is the timed trace obtained from λ' by forgetting the elements in $Act_{\mathcal{M}} \setminus J$ from the underlying

action sequence. A DTIOM $\mathcal{M}$ is said to be *feasible* if it is feasible in relation to $Act^I_{\mathcal{M}}$ and $\delta_{\mathcal{M}}$.

This notion of feasibility is similar to the one used, for example, in (Kaynar et al., 2006), which we have relativised to given sets of input actions in order to account for structured interactions with the environment.

Proposition 4.3. A DTIOM $\mathcal{M}$ is feasible in relation to $J \subseteq Act^I_{\mathcal{M}}$ and a multiple δ of $\delta_{\mathcal{M}}$ if it is initializable, makes independent progress and is DP-enabled in relation to J and δ.

In relation to the compositionality of feasibility, we can prove:

Theorem 4.2. Let $\mathcal{M}_1$ and $\mathcal{M}_2$ be δ-compatible DTIOMs that can make independent progress. Let $\mathcal{M} = {}_{\delta}\|\|_{i=1,2}^{(l_i,v_i,t_i)} \mathcal{M}_i$ where, for $i = 1, 2$, (l_i, v_i) is a state of $\mathcal{M}_i$ reachable at time t_i.

Let δ'_1 be a multiple of δ_1 such that t_1 is a multiple of δ'_1 and $J \subseteq Act^I_1$. If

1) $\mathcal{M}_1$ is DP-enabled in relation to J and δ'_1, and
2) for some δ' multiple of δ_1 and δ_2 such that t_1 and t_2 are multiples of δ',
   (a) $\mathcal{M}_1$ is DP-enabled in relation to $Act^I_1 \cap Act^O_2$ and δ'
   (b) $\mathcal{M}_2$ is DP-enabled in relation to $Act^I_2 \cap Act^O_1$ and δ'
   (c) both $\mathcal{M}_1$ and $\mathcal{M}_2$ are δ'-cooperative in relation to $Act_1 \cap Act_2$

then $\mathcal{M}$ is feasible in relation to $J \setminus Act^O_2$ and δ'_1.

5. Networks of timed machines

The systems that are now operating in cyberspace are best modelled as networks of machines, where each machine performs local computations and interacts with other machines to achieve some goal. A further aspect of those networks is that they are *dynamic*, i.e., they are not assembled at design time by a software engineer, but result from interconnections between existing networks established at *run time*, while they are executing, possibly performed by middleware components (e.g., in the context of service-oriented architectures (Papazoglou et al., 2007)). This ensures that systems can procure resources or services only when they are required and according to constraints that depend on current needs of those systems.

In this section, we provide a mathematical model for such dynamic networks of timed machines abstracting away from the way in which services or resources are procured (see (Fiadeiro et al., 2011) for a formalisation of service discovery, selection and binding). Our model is based on graphs whose nodes are labelled with localised and interfaced timed machines and whose edges are labelled with relations that establish how timed machines interact with each other.

5.1. *Localised timed machines*

In order to model the dynamic aspects of networks, we need to know the execution history of a given timed machine when it joins a network — the traditional, design-time (static) way of constructing systems takes components at their initial state. This motivates the following definition:

Definition 5.1 (Localised timed machine). A *localised timed machine* is a pair $\langle \mathcal{M}, \eta \rangle$ — denoted $\mathcal{M}@\eta$ — where $\mathcal{M}$ is a consistent DTIOM (in the sense of Def. 4.1) and η (the *history* of $\mathcal{M}$) is either a sequence consisting of the initial state of $\mathcal{M}$ — which we denote by $\eta_{\mathcal{M}}$ — or one of its partial executions starting at the initial state of $\mathcal{M}$.

We use the following composition operation on partial sequences: given a partial execution η and another partial execution η' whose first state is the last state of η, we denote by $\eta;\eta'$ the partial execution that follows from η by executing η'. If η' consists of only one state (the last state of η) then $\eta;\eta' = \eta$.

We denote by (l_η, v_η) the state of $\mathcal{M}$ reached at the end of η, and by t_η the (local) time at which that state was reached (0 in the case of $\eta_{\mathcal{M}}$). We ask that $\mathcal{M}$ be consistent so that, as a system, it can generate at least one trace (which could consist of the sequence of the empty set of actions, i.e., the machine might remain idle).

We are now interested in the behaviour of the timed machine from the designated reachable state; therefore, we adapt the notions of execution, behaviour and language given in Defs. 3.2 and 3.1 for DTIOMs:

Definition 5.2 (Execution, behaviour and language). The *executions* and *partial executions* of $\mathcal{M}@\eta$ are those of $\mathcal{M}$ that start at (l_η, v_η). The *behaviour* $[\![\mathcal{M}@\eta]\!]$ of $\mathcal{M}@\eta$ is the set of its executions and its *language* $\Lambda_{\mathcal{M}@\eta}$ is the r-closure of the set of timed traces defined by $[\![\mathcal{M}@\eta]\!]$.

Notice that $\mathcal{M}$ and $\mathcal{M}@\langle l^\star, v^\star \rangle$, where $l^\star, v^\star$ are the initial location and clock valuation of $\mathcal{A}_{\mathcal{M}}$, respectively, are essentially the "same" (even if, formally, they are different structures: the former is a machine and the latter is a localised machine).

The following is a trivial (but useful) property of executions of localised timed machines:

Proposition and Definition 5.1. Every trace $\lambda{=}\langle \sigma, \tau_{\delta_{\mathcal{M}}} \rangle{\in}\Lambda_{\mathcal{M}@\eta}$ defines the trace $\eta;\lambda{=}\langle \sigma_\eta;\sigma^1, \tau_{\delta_{\mathcal{M}}} \rangle{\in}\Lambda_{\mathcal{M}}$ where σ_η is the sequence consisting of $\emptyset$ followed by the sequence of sets of actions defined by η, and σ^1 is the suffix of σ starting at position 1.

Notice that we use the suffix σ^1 in the concatenation because, by Def. 2.1, every timed trace starts with the empty set of actions at time 0, which needs to be removed.

Corollary 5.1. Every trace $\lambda = \langle \sigma, \tau \rangle \in \Lambda_{\mathcal{M}@\eta}$ defines $\eta;\lambda = \langle \sigma_\eta;\sigma^1, \tau_\eta;\tau' \rangle \in \Lambda_{\mathcal{M}}$ where σ_η is as above, $\tau_\eta = \langle 0, \ldots, t_\eta \rangle$, and $\tau'(i) = \tau(i+1) + t_\eta$.

5.2. *Limitations of the algebra of timed machines*

The traditional approach to system modelling based on a component algebra uses the composition operator of that algebra to model the behaviour of a composite system. For example, in typical process algebras, the (parallel) composition of two processes is taken to provide a model of the joint execution of those processes according to a given protocol (for example, synchronisation of inputs with outputs). One of the major differences between the approach to systems that we are proposing in this paper and the traditional ones becomes apparent when we analyse parallel composition of machines in more detail, which we do in this section.

The composition of two TIOA as defined in Sec. 2.2 and, by extension, the *homogeneous* composition of two DTIOMs as defined in Sec. 3.2, can be taken as a model of a system of two components synchronising on their shared input/output pairs. This is because the language that results from the composition coincides with the intersection of the languages of the components, which is what is usually taken to be the joint behaviour of the system of the two components in a trace-based semantic domain.

More precisely, it is not difficult to prove that, if two machines $\mathcal{M}_1@\eta_1$ and $\mathcal{M}_2@\eta_2$ have the same granularity δ, the language $\Lambda_{\mathcal{M}}$ of the composition $\mathcal{M} = ||_{i=1,2}^{(l_{\eta_i}, v_{\eta_i}, t_{\eta_i})} \mathcal{M}_i$ is the intersection $\iota_1(\Lambda_{\mathcal{M}_1@\eta_1}) \cap \iota_2(\Lambda_{\mathcal{M}_2@\eta_2})$ where

— ι_i is the inclusion of Act_i in $Act_1 \cup Act_2$
— $\iota_i(\Lambda_{\mathcal{M}_i@\eta_i}) = \{\langle \sigma, \tau \rangle : \langle \sigma|_{\iota_i}, \tau \rangle \in \Lambda_{\mathcal{M}_i@\eta_i}\}$ where, for every k, $\sigma|_{\iota_i}(k) = \sigma(k) \cap Act_i$ (see Def. 2.3)

That is, the language of the parallel composition consists of the timed traces that project to timed traces of the languages of the machines.

If $\mathcal{M}_1@\eta_1$ and $\mathcal{M}_2@\eta_2$ have different clock granularities δ_1 and δ_2, respectively, we can still calculate $\iota_1(\Lambda_{\mathcal{M}_1@\eta_1}) \cap \iota_2(\Lambda_{\mathcal{M}_2@\eta_2})$, which is the joint behaviour of the two machines synchronising on shared inputs and outputs at times that are multiple of both δ_1 and δ_2. If no such multiples exist, the two machines cannot synchronise and, therefore, either they do not have liveness requirements (in the sense that they are not required to eventually do something, i.e., their languages include timed traces that only execute the empty set of actions) or they cannot agree on any timed trace — their interconnection is inconsistent.

If δ_1 and δ_2 are commensurate, i.e., admit a common divisor δ, we saw in Sec. 3.2 how, through the notion of refinement, we can define the heterogeneous composition of two compatible DTIOMs. However, the language of a heterogeneous composition is not necessarily the intersection of the languages of the components. For example, the former will consist of traces that refine the granularity defined by the common divisor of the clock granularities of the two timed machines, whereas the latter may have traces that do not conform to that time pattern. In this sense, *parallel composition of machines does not capture the joint behaviour of the system consisting of those machines.*

Nevertheless, if the machines are compatible, their parallel composition is a best approximation of their joint behaviour:

Theorem 5.1. Let $\mathcal{M}_i@\eta_i$ – $i = 1, 2$ – be such that $\mathcal{M}_1$ and $\mathcal{M}_2$ are compatible and

let δ be the greatest common divisor of their clock granularities. The composition

$$\mathcal{M} = \|_{\delta}^{(l_{\eta_i}, v_{\eta_i}, t_{\eta_i})}_{i=1,2} \mathcal{M}_i$$

is the machine that best approximates $\Lambda = \iota_1(\Lambda_{\mathcal{M}_1@\eta_1}) \cap \iota_2(\Lambda_{\mathcal{M}_2@\eta_2})$, i.e.,

— $\Lambda_{\mathcal{M}} \precapprox \Lambda$ and,
— for any other machine $\mathcal{M}'$ such that $\Lambda_{\mathcal{M}'} \precapprox \Lambda$, $\mathcal{M}' \precapprox \mathcal{M}$.

Having a best approximation is important so that properties of the joint behaviour of the two localised timed machines can be inferred from or simulated by that of the composite machine. For example, this result is important to certify that the behaviour Λ of a system of components that implement given compatible machines $\mathcal{M}_i@\eta_i$ – $i = 1, 2$ – is not empty and, hence, that the components can indeed operate together. This is because, by Theo. 5.1, $\mathcal{M} = \|_{i=1,2}^{(l_{\eta_i}, v_{\eta_i}, t_{\eta_i})} \mathcal{M}_i$ is such that $\Lambda_{\mathcal{M}} \precapprox \Lambda$ and, hence, if $\mathcal{M}$ is consistent, Λ is not empty.

This is the starting point for the network algebra that we define in the remainder of this section.

5.3. *Interfaced machines, attachments and networks*

In the previous section, we have shown that the machine that results from applying the composition operator to two machines only approximates their joint behaviour. That is, the algebra of timed machines that we developed in Sec. 3.2 does not provide an exact model for systems of machines.

In the context of modern systems (of systems) whose structure change as they interconnect, dynamically, with other systems, reducing a system to a (big) component does not make sense either: where in traditional software engineering a system is delivered as a whole to a client, networks of systems that operate in cyberspace are not delivered to anyone — they behave autonomously. That is, we are interested in networks as objects.

This is why we develop instead an algebra of *networks* of localised timed machines that can capture the dynamics of networks and over which we can reason about the joint behaviour of the machines that execute over the network; the composition operator of this network algebra does not compose machines — it interconnects them without reducing them to a single machine.

In order to be able to be interconnected in networks, machines need interfaces through which they can expose the actions (input or output) through which they can interact with other machines. For that purpose, we organise actions in sets that we call ports: a *port* is a finite non-empty set (of actions).

Definition 5.3 (Interfaced timed machine). An *interfaced DTIOM* is a pair $\langle \mathcal{M}, \gamma \rangle$ where

— $\mathcal{M} = \langle \delta, \mathcal{A} \rangle$ is a DTIOM;
— γ is a finite set of mutually disjoint ports such that $Act^I_{\mathcal{A}} \cup Act^O_{\mathcal{A}} = \bigcup_{M \in \gamma} M$.

That is, each input and each output action of the machine belongs to exactly one of its

ports. Notice that ports may have a mix of input and output actions. This is because ports are abstractions that are convenient for organising networks of machines (as formalised below) where machines exhibit, at each port, interactions that may be more complex than a simple input/output relation.

Because we want that machines can bind to other machines at run time, not design time, it would not be realistic to assume that machines have mutually disjoint alphabets. Therefore, we need a way of interconnecting machines that is not based on shared names, i.e., we need explicit, not implicit interconnections.

Definition 5.4 (Attachment). An *attachment* between $\langle \mathcal{M}_1, \gamma_1 \rangle$ and $\langle \mathcal{M}_2, \gamma_2 \rangle$ is a bijection $\xi : M_1 \leftrightarrow M_2$ – $M_1 \in \gamma_1$ and $M_2 \in \gamma_2$ – that reverses polarities, i.e., if $a_1 \xi a_2$ then $a_1 \in Act_1^I$ iff $a_2 \in Act_2^O$.

That is, an attachment establishes a one-to-one correspondence between two ports such that any two actions that are connected have opposite I/O polarities.

Interfaced timed machines can be localised in exactly the same way as timed machines (Def. 5.1), for which we use the notation $\langle \mathcal{M}, \gamma \rangle @ \eta$. In the rest of the paper, we often refer to localised interfaced timed machines simply as 'timed machines' or 'machines'.

We can now define networks of (localised interfaced) timed machines as follows:

Definition 5.5 (Network of timed machines). A *network of localised interfaced timed machines* (NTM) α consists of:

— A finite undirected (multi)graph $\langle N, E \rangle$ where N is a non-empty finite set (of nodes) and E is a finite set (of edges) equipped with a function that associates an unordered pair $\{p, q\}$ of distinct nodes to every edge e (which we normally denote $e : \{p, q\}$). That is, we allow for more than one edge between any two nodes but no edge between any node and itself.

— A labelling function that assigns to every node p a localised interfaced timed machine α_p and, to every edge $e : \{p, q\}$, an attachment ξ_e between α_p and α_q such that:

  (a) Edges only connect machines with commensurate time granularities, i.e., if $e : \{p, q\}$ then α_p and α_p have commensurate time granularities.

  (b) No port is involved in more than one attachment, i.e., if $e_1 : \{p, q_1\}$ and $e_2 : \{p, q_2\}$ with $q_1 \neq q_2$ then the port of α_p that is used in ξ_{e_1} is different from that used in ξ_{e_2}.

For every node p, we use $\mathcal{M}_p = \langle \delta_p, \mathcal{A}_p \rangle, \gamma_p$ and η_p to denote the various components of α_p, and Act_p and Λ_p to denote the set of actions of $\mathcal{A}_p$ and the language of $\mathcal{M}_p @ \eta_p$, respectively.

We also define the following sets and mappings:

— Act_α is the set (alphabet) of actions associated with α, which is obtained as follows: for every $p \in N$ and every $M \in \gamma_p$,

  – if there is an edge $e = \{p, q\}$ and an attachment ξ_e between α_p and α_q that connects M then, for every action $a \in M$, $\{p.a, q.\xi_e(a)\} \in Act_\alpha$;

  – otherwise, i.e., if M is not connected to any other node of the graph, for every action $a \in M$, $p.a \in Act_\alpha$.

That is, we prefix all actions with the node from which they originate unless the actions belong to a port that is connected in the graph, in which case the actions are also paired with the actions to which they are attached to form synchronisation sets.

— For every $p \in N$, ι_p is the function that maps Act_p to Act_α, which prefixes the actions of Act_p as described above.

— $\Lambda_\alpha = \{\lambda \in \Lambda(Act_\alpha) : \forall p \in N(\lambda|_{\iota_p} \in \Lambda_p)\}$.

— $\Pi_\alpha = \{\pi \in \Pi(Act_\alpha) : \forall p \in N(\pi|_{\iota_p} \in \Pi_p)\}$.

Note that Act_α is well defined because no port is involved in more than one attachment and, for any attachment $\xi : M_1 \leftrightarrow M_2$ associated with an edge $e = \{p_1, p_2\}$, the corresponding synchronisation sets are of the form $\{p_1.a_1, p_2.\xi_e(a_1)\}$, which are the same as $\{p_2.a_2, p_1.\xi_e^{-1}(a_2)\}$ if $a_2 = \xi_e(a_1)$.

It is important to note that the nodes of the graph *are not* machines: they are *labelled with* machines. This means that several nodes may be labeled with the same machine (several instances of the same machine are running autonomously in the network), or with the same DTIOM but with different localisations (meaning that they joined the network at different stages of execution), or with machines that share the same TIOA but operate on a different clock granularity (meaning that the network connects devices of the same kind but operating over different platforms). In a sense, machines act as types and the nodes act as instances of the types that label them. This is reflected in the use of the translations ι that prefix every action with the node to which they belong.

Also notice that, for every $p \in N$, $(\_|_{\iota_p})$ first removes the actions that are not in the image of the language Act_p by ι_p, and then removes the prefix; in the case of synchronisation sets, it also removes the action that belongs to the other node. Therefore, the set Λ_α consists of all traces over the alphabet of the network that are projected to traces of the languages of the timed machines in the network:

$$\Lambda_\alpha = \bigcap_{p \in N} \iota_p(\Lambda_p)$$

We take this set to represent the behaviour of α. That is, the behaviour of the network is given by the intersection of the behaviour of the timed machines translated to the language of the network, which is how we defined the behaviour of the interconnection of two timed machines in Sec. 5.2.

Notice that, because the automata involved are open, the translations applied to sets of traces effectively open the behaviour of the machines to actions in which they are not involved. Furthermore, because the languages of machines are r-closed, so is Λ_α (the intersection of r-closed sets being r-closed), which is why we are able to deal with different clock granularities.

A particular case of a network (see Fig. 7) is a graph with two nodes labelled with the DTIOMs $\mathcal{M}^x$ and $\mathcal{M}^y$ of Ex. 3.3, each with a single interface — Act^x and Act^y, respectively — and a single edge labelled with the identity $\{a, b\} \leftrightarrow \{a, b\}$ and localised at their initial states, which is well formed because δ^x and δ^y have commensurate clock granularities and the identity reverses polarities. As discussed in Sec. 5.2 (Theo. 5.1), the

machine $\mathcal{M}^x \,||^{(A,0,0),(1,0,0)}\, \mathcal{M}^y$ is a best approximation of the behaviour of the network, i.e., of $\Lambda_{\mathcal{M}^x@(A,0)} \cap \Lambda_{\mathcal{M}^y@(1,0)}$.

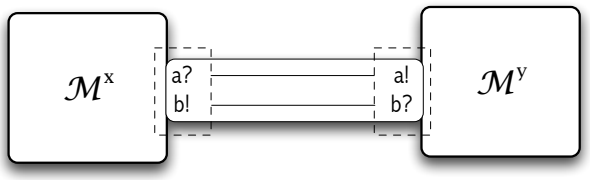

Fig. 7. A network consisting of two nodes, one labelled with $\mathcal{M}^x$ and the other with $\mathcal{M}^y$, and an edge labelled with $\{a, b\} \leftrightarrow \{a, b\}$ connecting the two nodes.

As also discussed in Sec. 5.2, we cannot necessarily find a machine $\mathcal{M}$ that generates Λ_α as its language: because the different clock granularities in α do not necessarily agree with each other, it is possible to have timed traces $\langle \sigma, \tau \rangle$ in Λ_α for which there is no clock granularity δ such that τ is a δ-time sequence; on the contrary, all timed traces generated by a machine have a time granularity (that of its clock).

Definition 5.6 (Approximation). We say that a machine $\mathcal{M}$ is an approximation of a network α ($\mathcal{M} \precapprox \alpha$) if $\Lambda_\mathcal{M} \precapprox \Lambda_\alpha$. A machine $\mathcal{M}_\alpha$ is a best approximation of α if $\mathcal{M}_\alpha \precapprox \alpha$ and, for any other machine $\mathcal{M}$ such that $\mathcal{M} \precapprox \alpha$, $\mathcal{M} \precapprox \mathcal{M}_\alpha$.

Proposition 5.1. If $\mathcal{M} \precapprox \alpha$ then $\delta_\mathcal{M}$ is a common divisor of all the clock granularities of the nodes of α.

Therefore, if no common divisor of all the clock granularities of the nodes of α exists, α cannot be approximated by a machine.

We can generalise Theo. 5.1 to the case where a common divisor of all the clock granularities of its nodes exists. In order to compute a machine that (best) approximates α, we refine all the machines of α to operate over the greatest common divisor δ_α.

Theorem 5.2. Let α be a network such that a common divisor of all the clock granularities of its nodes exists and δ_α be their greatest common divisor. The machine

$$\mathcal{M} = {}_{\delta_\alpha}\Big|\Big|_{p \in N_\alpha}^{(l_{\eta_p}, v_{\eta_p}, t_{\eta_p})} \iota_p(\mathcal{M}_p)$$

where the ι_p translate the alphabet of the automata α_p to Act_α (Def. 5.5) and prefix all clocks of the automata with $p.$, is the best approximation of α.

A particular case is when α is connected, i.e., when there is a path between any two of its nodes.

Proposition 5.2. For every connected network α, there is a common divisor of the clock granularities of the machines that label its nodes.

Corollary 5.2. For every connected network α, there is a machine $\mathcal{M}_\alpha$ that best approximates it.

This result is useful to determine how a connected component of a network can be executed (or simulated) over a processor (or set of processors sharing the same clock). It will also be useful in the next section to reason about consistency of networks.

5.4. *Dynamic composition of networks*

A network consisting of just two machines with no interconnection between them can hardly be meaningfully considered as a 'system': their behaviours do not interfere with each other and, hence, no emergent behaviour would result from considering them together. Therefore, the networks that are of interest as building blocks of systems are those that are connected. In this section, we provide an algebra of connected networks whose composition operator interconnects two separate connected networks or adds an interconnection within an existing connected network. To extend this algebra to arbitrary networks one would just need another operation of juxtaposing two networks. In Sec. 5.5, we also discuss how important properties of networks can be reduced to those of their connected components.

In order to establish interconnections, we need the notion of interaction-point, i.e., ports that are available to be interconnected:

Definition 5.7 (Interaction-point). An interaction-point of a network α is a pair $\langle p, M\rangle$ such that $p{\in}N_\alpha$ and $M{\in}\gamma_p$ is not used in any edge $e : \{p, q\}$ of α — we denote by J_α the collection of interaction-points of α. For every interaction point $\langle p, M\rangle$, we define $\langle p, M\rangle^I = p.(M \cap Act_p^I)$ and $\langle p, M\rangle^O = p.(M \cap Act_p^O)$ — the input and output actions that belong to the interface M prefixed with $p.$, respectively.

The following constructors can be defined over networks:

Definition 5.8 (Atomic network). We build an atomic network with a single node labelled with an interfaced timed machine localised in its initial state.

Formally, given a consistent interfaced timed machine $\langle \mathcal{M}, \gamma\rangle$, we build the network $\alpha_{\langle \mathcal{M},\gamma\rangle}$ as follows:

— Its graph is $\langle\{p\}, \emptyset\rangle$.
— Its labelling function assigns $\langle \mathcal{M}, \gamma\rangle @ \eta_{\mathcal{M}}$ to p.

Because, as already argued, $\langle \mathcal{M}, \gamma\rangle$ and $\alpha_{\langle \mathcal{M},\gamma\rangle}$ are essentially the same, this operation simply turns a consistent interfaced timed machine into a network.

Definition 5.9 (Binding operations). We define two constructors that add attachments: one between two networks and the other within a network.

inter-binding: We attach an interaction-point of a connected network to an interaction-point of another connected network. Because the machines associated with the corresponding nodes will need to interact via their interaction points, their clock granularities need to be commensurate. In the resulting network, the histories of all the machines are updated with the partial executions that they will have performed at the time of the binding.

Formally, given:

— two connected networks $\langle N_i, E_i, \alpha_i, \xi_i \rangle$, $i = 1, 2$;

— an attachment $\xi_{\{q_1,q_2\}} : M_{q_1} \leftrightarrow M_{q_2}$ between two interaction-points where $q_i \in N_i$ and δ_{q_1} and δ_{q_2} are commensurate;

— for each $i = 1, 2$, a δ_{α_i}-time prefix π_{α_i} of Π_{α_i} (the prefix of α_i at the time of the binding) and, for each $p \in N_{\alpha_i}$, a partial execution η'_p of $\mathcal{M}_p$ whose first state is the last state of η_p and such that the projection of π_{α_i} to the behaviour of each $\mathcal{M}_p$ refines the prefix generated by η'_p (η'_p is the partial execution performed by $\mathcal{M}_p$ as part of α_i);

the result of the inter-binding is the network $(\alpha_1 \ {}_{\xi_{\{q_1,q_2\}}} \ \alpha_2)$ defined as follows:

— Its set of nodes is the disjoint union of the sets of nodes of the two networks.

— Its set of edges is the disjoint union of the sets of edges of the two networks plus an additional edge $e : \{q_1, q_2\}$ connecting the two interaction points.

— Its node-labelling function assigns to every node $p_i \in N_i$ the same machine $\mathcal{M}_{p_i}$ but localised at $\eta_{p_i}; \eta'_{p_i}$ (the history of $\mathcal{M}_{p_i}$ at the time the binding is established).

— Its edge-labelling function assigns to every edge $e_i \in E_i$ the same attachment $\xi_{i_{e_i}}$, and assigns to the new edge $e : \{q_1, q_2\}$ the attachment $\xi_{\{q_1,q_2\}}$.

intra-binding: Given a connected network, we add an attachment between two of its interaction points (which are necessarily labelled with machines that have commensurate clock granularities because the network is connected). In the resulting network, the histories of all the machines are updated with the partial executions that they will have performed at the time of the binding.

Formally, given:

— a connected network $\langle N, E, \alpha, \xi \rangle$;

— an attachment $\xi_{\{q_1,q_2\}} : M_{q_1} \leftrightarrow M_{q_2}$ between two of its interaction points;

— a δ_α-time prefix π of Π_α (the prefix at the time of the binding) and, for each $p \in N$, a partial execution η'_p of $\mathcal{M}_p$ whose first state is the last state of η_p and such that the projection of π to the behaviour of each $\mathcal{M}_p$ refines the prefix generated by η'_p;

the result of the intra-binding is the network $\alpha^+_{\xi_{\{q_1,q_2\}}}$ defined as follows:

— Its set of nodes is N.

— Its set of edges is $E \cup \{e\}$ where $e \notin E$ and $e : \{q_1, q_2\}$.

— Its node-labelling function assigns to every node $p \in N$ the same machine $\mathcal{M}_p$ but localised at $\eta_p; \eta'_p$.

— Its edge-labelling function is identical to ξ on E and assigns to the new edge $e : \{q_1, q_2\}$ the attachment $\xi_{\{q_1,q_2\}}$.

Notice that, as a result of a binding, two of the interaction-points are lost, i.e., no more interconnections can be made through the ports that were used. This achieves at the network level what internal actions do at the machine level by removing input/output pairs from the sets of input and output actions of machines that are composed in parallel. Interconnecting machines in a network does not compose them in an algebraic sense (i.e., a new machine is not returned), but the ports that are interconnected are no longer available for other interconnections.

Binding creates a new network, even in the case of intra-binding because of the addition of a new attachment. This is why it is important that timed machines are localised in networks: to determine the behaviour of the new network, we need to know the history of each of the machines involved at the time of the binding as this will determine how they are going to progress within the new context of interactions that the network creates.

The fact that the networks are connected, and that the intra-binding is performed through attachments between machines whose clock granularities are commensurate (which ensures that the new network is well defined), ensures that the binding can be made when all the machines involved have reached a state, i.e., at a multiple of their clock granularities. Therefore, given a trace of the new network, we can reconstruct a trace of each of the machines involved, which ensures that machines are indeed autonomous, i.e., their behaviour is fully determined by the TIOA according to which they execute, their clock granularities, and the interactions they have with other machines. Indeed, using Cor. 5.1, we can establish the following two results:

Proposition 5.3. Let α' be the result of inter-binding α_1 and α_2.

(1) Let $p_i \in N_{\alpha_i}$ and η'_{p_i} be the partial execution of $\mathcal{M}_{p_i}$ at which the binding is made. Given a trace $\lambda' \in \Lambda_{\alpha'}$, $(\eta_{p_i}; \eta'_{p_i}); \lambda'|_{\iota_{p_i}} \in \Lambda_{\mathcal{M}_{p_i}}$.

(2) Let $\pi_{\alpha_i} \in \Pi_{\alpha_i}$ be the prefix of α_i at which the binding is made and $\iota_i : Act_{\alpha_i} \rightarrow Act_{\alpha'}$ the mappings between their two alphabets. Given $\lambda' \in \Lambda_{\alpha'}$, $\pi_{\alpha_i}.\lambda'|_{\iota_i} \in \Lambda_{\alpha_i}$.

Proposition 5.4. Let α' be the result of intra-binding within α.

(1) Let $p \in N_\alpha$ and η'_p be the partial execution of $\mathcal{M}_p$ at which the binding is made. Given a trace $\lambda' \in \Lambda_{\alpha'}$, $(\eta_p; \eta'_p); \lambda'|_{\iota_p} \in \Lambda_{\mathcal{M}_p}$.

(2) Let $\pi \in \Pi_\alpha$ be the prefix of α at which the binding is made and $\iota : Act_\alpha \rightarrow Act_{\alpha'}$ the mapping between their two alphabets. Given $\lambda' \in \Lambda_{\alpha'}$, $\pi.\lambda'|_\iota \in \Lambda_\alpha$.

That is, any behaviour of a machine after a binding is in the language of the machine given the history at the point of the binding. Furthermore, every trace that is generated after the binding is an admissible continuation of the prefixes at the time of the binding; this is because, although the synchronisation that results from the binding was not originally required, it was admissible. Notice that inter- and intra-binding can only take place at a time that is a multiple of the clock granularities of all the machines involved in the networks.

Naturally, it still remains to investigate under which conditions we can guarantee that the new network can generate any behaviour at all, i.e., that the machines can interoperate according to the new interconnection, which we do in the next section.

One can also define operations that unbind (destructors), i.e., remove an attachment, which either produce a network if the removal of the attachment preserves connectivity, or two networks otherwise.

Definition 5.10 (Unbinding operations). Given:

— a connected network $\langle N, E, \alpha, \xi \rangle$;
— an edge $e : \{q_1, q_2\}$;
— a prefix π of Π_α at the time of the unbinding and, for each $p \in N$, a partial execution η'_p of $\mathcal{M}_p$, such that the projection of π to the behaviour of each $\mathcal{M}_p$ is generated by η'_p;

we define the network α' as follows:

— Its set of nodes is N.
— Its set of edges is $E \setminus \{e\}$.
— Its node-labelling function assigns to every node $p \in N$ the same machine $\mathcal{M}_p$ but localised at $\eta_p; \eta'_p$.
— Its edge-labelling function is identical on $E \setminus \{e\}$.

The result of the unbinding is:

— α' if the network is connected, which we denote by α^-_e;
— otherwise, the two connected components that result from removing the edge, which we denote by $\alpha^-_{e_{q_1}}$ and $\alpha^-_{e_{q_2}}$.

Naturally, $\alpha^-_{e_{q_1}}$ and $\alpha^-_{e_{q_2}}$ can be considered as the connected components of the network obtained by simply juxtaposing them.

We can also prove:

Proposition 5.5. Let α' result from unbinding within α.

(1) Let $p \in N_{\alpha'}$ and η'_p be the partial execution of $\mathcal{M}_p$ at which the unbinding is made. Given a trace $\lambda' \in \Lambda_{\alpha'}$, $(\eta_p; \eta'_p); \lambda'|_{\iota_p} \in \Lambda_{\mathcal{M}_p}$.
(2) Let $\pi \in \Pi_\alpha$ be the prefix of α at which the unbinding is made and $\iota : Act_{\alpha'} \to Act_\alpha$ the mapping between their two alphabets (which reflects the removal of the attachment). Given $\pi.\lambda \in \Lambda_\alpha$, $\lambda|_\iota \in \Lambda_{\alpha'}$.

That is, like for binding, any behaviour of a machine after unbinding is in the language of the machine given the history at the point of the binding. However, the result is now that any suffix that could have been generated in the original network after unbinding is an admissible trace of the new network.

5.5. *Consistency and feasibility*

We now generalise the notions of consistency (Def. 4.1) and feasibility (Def. 4.6) to networks.

Definition 5.11 (Consistent network). A network α is *consistent* if $\Lambda_\alpha \neq \emptyset$.

Consistency means that there is a timed trace that is shared by all timed machines

(nodes) as connected through the attachments (edges). A stronger property requires that, no matter what inputs the network receives from its environment at an interaction-point, it can produce a joint timed trace.

Definition 5.12 (Feasibility). Let α be a network. The network α is said to be *feasible in relation to* an interaction-point $\langle p, M\rangle$ and a multiple δ of δ_p if, for every δ-timed trace λ over $\langle p, M\rangle^I$, there is a trace $\lambda' \in \Lambda_\alpha$ such that $\lambda'|_{\langle p,M\rangle^I} \preceq \lambda$, where $\lambda'|_{\langle p,M\rangle^I}$ is the timed trace obtained from λ' by forgetting the elements in $Act_\alpha \setminus \langle p, M\rangle^I$ from the underlying action sequence.

A network α is *feasible* if it is feasible in relation to J_α (the collection of all its interaction-points) and the corresponding clock granularities.

Returning to the discussion at the beginning of Sec. 5.4, consistency and feasibility are only challenging properties to prove for connected components of networks: two networks that are brought together as a network without any interconnection added to them will be consistent (resp. feasible) if and only if the component networks are. That is, the consistency or feasibility of a network can always be reduced to the consistency or feasibility of their connected components.

Therefore, we restrict the checking of consistency and feasibility to connected networks. We do so by extending the notions introduced in Sec. 4 to connected networks using Theo. 5.2, i.e., by using properties of the timed machine $\mathcal{M}_\alpha$ that is a best approximation of α.

Definition 5.13. Let α be a connected network.

— α is said to make independent progress iff $\mathcal{M}_\alpha$ makes independent progress.
— α is said to be DP-enabled in relation to one of its interaction-points $\langle p, M\rangle$ iff $\mathcal{M}_\alpha$ is DP-enabled in relation to $\langle p, M\rangle^I$ and δ_p (note that δ_p is a multiple of δ_α).
— α is said to be cooperative in relation to one of its interaction-points $\langle p, M\rangle$ and a multiple δ of δ_p, iff $\mathcal{M}_\alpha$ is cooperative in relation to $\langle p, M\rangle^O$ and δ.

The following theorem, which is the counterpart of Prop. 4.2 for networks, results directly from the fact that, for any connected network α, $\mathcal{M}_\alpha \precapprox \alpha$ (Theo. 5.2).

Proposition 5.6. If a connected network α makes independent progress, then it is consistent. If α is also DP-enabled in relation to each of its interaction-points, then it is feasible.

Notice that, because the machines that operate in any network α need to be consistent, $\mathcal{M}_\alpha$ is necessarily initializable. We consider now how the other properties can be checked in a compositional way for the constructors (atomic network and binding operations).

The checking of consistency and feasibility of atomic networks reduces to checking those properties over the DTIOM that labels the node, which has been covered in Sec. 4.

We now investigate how consistency and feasibility of networks obtained through binding can be checked. We first check properties of a network built through intra-binding:

Theorem 5.3. Let α be a connected network and $\xi_{\{p,q\}} : M_p \leftrightarrow M_q$ be an attachment where $\langle p, M_p\rangle$ and $\langle q, M_q\rangle$ are interaction points such that δ_p and δ_q are commensurate.

(1) If α makes independent progress and is DP-enabled in relation to the interaction points $\langle p, M_p\rangle$ and $\langle q, M_q\rangle$ and, for some δ common multiple of δ_p and δ_q, α is both δ-cooperative in relation to $\langle p, M_p\rangle$ and $\langle q, M_q\rangle$, then $\alpha^+_{\xi_{\{p,q\}}}$ also makes independent progress and, therefore, is consistent.

Moreover, for every interaction point $\langle v, M_v\rangle$ of α different from those involved in the attachment:

(2) If α is DP-enabled in relation to $\langle v, M_v\rangle$ so is $\alpha^+_{\xi_{\{p,q\}}}$

(3) If α is cooperative in relation to $\langle v, M_v\rangle$ and δ multiple of δ_v so is $\alpha^+_{\xi_{\{p,q\}}}$.

Therefore, under the conditions of (1) and (2) above, we can conclude that, for every interaction point $\langle v, M_v\rangle$ of α different from those involved in the attachment, $\alpha^+_{\xi_{\{p,q\}}}$ is feasible in relation to $\langle v, M_v\rangle$ if so is α.

Therefore, adding an attachment to a connected network does not change the sufficient conditions that determine the consistency and feasibility of the network.

In summary, the properties of making independent progress, being cooperative and being DP-enabled can be checked compositionally for connected networks, reducing the proofs of having those properties to the atomic cases (which are checked over simple machines).

Finally, we check networks built through inter-binding:

Theorem 5.4. Let α_1 and α_2 be two disjoint connected networks and $\xi_{\{p,q\}} : M_p \leftrightarrow M_q$ be an attachment where $\langle p, M_p\rangle$ is an interaction point of α_1 and $\langle q, M_q\rangle$ an interaction point of α_2 such that δ_p and δ_q are commensurate.

(1) If α_1 and α_2 make independent progress, α_1 is DP-enabled in relation to $\langle p, M_p\rangle$, α_2 is DP-enabled in relation to $\langle q, M_q\rangle$, and for some δ common multiple of δ_p and δ_q, α_1 is cooperative in relation to $\langle p, M_p\rangle$ and δ, and α_2 is cooperative in relation to $\langle q, M_q\rangle$ and δ, then $(\alpha_1\ {}_{\xi_{\{p,q\}}}\ \alpha_2)$ also makes independent progress and, therefore, is consistent.

Moreover, for every interaction point $\langle v, M_v\rangle$ of α_i different from that involved in the attachment:

(2) If α_i is DP-enabled in relation to $\langle v, M_v\rangle$ so is $(\alpha_1\ {}_{\xi_{\{p,q\}}}\ \alpha_2)$.

(3) If α_i is cooperative in relation to $\langle v, M_v\rangle$ and δ multiple of δ_v so is $(\alpha_1\ {}_{\xi_{\{p,q\}}}\ \alpha_2)$.

Therefore, under the conditions of (1) and (2) above, we can conclude that, for every interaction point $\langle v, M_v\rangle$ of α_i different from that involved in the attachment, $(\alpha_1\ {}_{\xi_{\{p,q\}}}\ \alpha_2)$ is feasible in relation to $\langle v, M_v\rangle$ if so is α_i.

In relation to unbinding, the following property is a corollary of Prop. 5.5.2:

Theorem 5.5. Let α' be a network that results from unbinding within α.

— α' is consistent if so is α.

— α' is feasible in relation to $\langle p, M_p\rangle$ and δ multiple of δ_p if so is α.

Notice that one cannot infer feasibility in relation to the new interaction points that are created by the unbinding. To check feasibility in relation to those points one needs to go

back to the properties of the individual machines in the network using the results proved above for binding.

6. Related Work

In this section, we discuss related work on models of dynamic reconfigurable systems, after which we cover other work that addresses heterogeneous time. Before that, we relate the work reported in this paper with our own previous work.

The model we propose in this paper is based on the algebra for timed-machines presented in (Delahaye et al., 2014), which encompasses two operations over timed machines: heterogeneous composition and refinement; this algebra was itself a generalisation of the homogeneous timed component algebra that was proposed in (Delahaye et al., 2013) for services. That algebra was extended herein with a new operator for the heterogeneous composition of timed machines that are not in their initial states to account for run-time interconnections of systems. This extension supports the definition of the new algebra of networks, which includes operations through which a network can be modified at run time through the binding of its components to components of other networks, intra-binding and unbinding.

A different algebra for heterogeneous networks of timed systems is presented in (Fiadeiro and Lopes, 2017) that is more abstract in the sense that is based purely on traces, i.e., it is independent of the nature of the machines that execute in the networks. That algebra is based on an asynchronous communication model and multi-party interactions. A logic is also proposed for specifying and reasoning about network behaviour.

Models for Dynamic Reconfiguration. Most of the work that addresses the modelling of systems with a structure that can change at run time follows a process-calculi approach. However, there have been a few attempts to develop state-based formalisms that handle dynamicity.

A mathematical model for dynamic systems based on an extension of I/O automata is presented in (Attie and Lynch, 2001; Attie and Lynch, 2016). Dynamic I/O Automata support the dynamic creation of automata and the dynamic change in the communication links (represented by changes in the signatures). The possible evolutions of a dynamic network of interacting automata are defined at design-time and, hence, its semantics is itself captured in terms of a (configuration) automaton.

A similar perspective is taken in (Fisher et al., 2011), which proposes Dynamic Reactive Modules – a generalisation of Reactive Modules (Alur and Henzinger, 1999) – to support the dynamic reconfiguration and creation/death of new processes by adapting concepts from object-oriented programming languages to the world of state-transition systems communicating by shared variables. More specifically, the mechanisms of object creation from classes and referencing are adapted to modules and processes: module classes can be instantiated, values can be passed during instantiation, and a reference to any newly created module is returned so that it can communicate with the other modules and vice versa.

Another state-based formalism that supports dynamic reconfiguration is Dynamic BIP

(Bozga et al., 2012). This formalism extends the BIP component framework (Basu et al., 2011) in order to support the description of architectures that admit dynamic change of interactions among a (fixed) set of components. In contrast with the two other approaches mentioned above, BIP enforces a clear separation between the behaviour of the individual elements and the way they are interconnected. However, Dynamic BIP blurs this separation: interactions are not explicitly described but, instead, need to be calculated for each state (based on the current state and the declared local constraints).

As far as we are aware, ours is the first framework that offers a component algebra for run-time interconnection of systems that does not assume that the way the network of systems can evolve is known at design time and is somehow executed from within the systems. Our mathematical model for dynamic networks of systems abstracts away from the reasons that, for instance, determine that a running system joins or leaves a network at a given point in time. Models that capture this type of decisions can be built upon our basic model, be it a central controller as in Dynamic I/O Automata or object-oriented mechanisms as in Dynamic Reactive Modules; yet another model could be based on service-oriented architectures, i.e., by building on service discovery, selection and binding mechanisms, which we have explored in (Fiadeiro and Lopes, 2010), though using a different component algebra.

Heterogeneous Time. Several researchers have addressed discrete timed systems with heterogeneous clock granularities. However, the main focus has been either on specification or on modelling and simulation, not so much on the challenges that heterogeneity raises on run-time interconnection of systems. For example, Forget et al. propose in (Forget et al., 2008) a synchronous data-flow language that supports the modelling of multi-periodic systems. In this setting, each system has its own discrete periodic clock granularity; composition is supported by a formal clock calculus that allows, in particular, for the refinement of clock granularities in a way that is similar to what we propose in Sec. 3. Aside from the fact that we adopt an automata-based representation, the main difference with our work is that they leave open the question of component-based verification of properties such as consistency.

Similarly, in (Chen et al., 2015), the authors introduce a formal communication model of behaviour for the composition of heterogeneous real-time cyber-physical systems based on logical clock constraints. Although their model supports the combination of heterogeneous timed systems, the authors do not consider the particular case of discrete periodic systems. In (Sander and Jantsch, 2004), the authors present a methodology (ForSyDe) for high-level modelling and refinement of heterogeneous embedded systems; whilst the semantics they propose, and the notion of clock-refinement they introduce, are similar in essence to ours, their main focus is again on modelling and simulation, whereas ours is on the structures that support compositional reasoning over properties of interconnected systems.

To cope with heterogenous time scales, several approaches to the specification of real-time systems, notably the Timebands Framework (Burns and Hayes, 2010), have also adopted an explicit representation of time granularity. That framework, unlike others, does not require that all descriptions be transformed into the finest granularity.

Heterogeneous time scales are also addressed in (Bliudze and Krob, 2009) in the context of a unified framework for discrete and continuous systems that adopts an approach to modelling time similar to ours. Systems are regarded as dataflow transformers where dataflows can be observed at any time but their values allowed to change only at discrete moments specified by the underlying clock. Periodic clocks with a fixed period δ are identified therein as the most important class of clocks for modelling real systems. The challenge of data synchronisation between different time scales is addressed by considering that machines have separate clocks for input, output and internal operations (constraining when inputs can be received, outputs can be produced and internal transitions can be performed, respectively). This work was generalised in (Golden et al., 2012) by doing away with the clock periodicity and considering instead one-slot buffers to synchronise dataflows.

Some attempts have also been made at addressing compositionality, for example in (Le et al., 2013) through the concept of tag machine (Benveniste et al., 2005). However, the notion of composition of systems introduced by the authors (using tag morphisms) is more relaxed than ours in that it allows for the delay between events to be modified in given tag machines. A consequence of this generality is that the language resulting from a composition is not an approximation of the intersection of the original languages, which, as argued in our paper, is essential for addressing global properties of interconnected systems as implemented.

From a practical point of view, some tools have been developed for modelling and simulating heterogeneous systems. For example, Ptolemy Classic (Buck et al., 1994) introduced the concept of heterogeneous combinations of semantics such as asynchronous models with timed discrete-events models. The concept was picked up in other tools such as System C (Grötker, 2002), Metropolis (Gößler and Sangiovanni-Vincentelli, 2002) and Ptolemy II (Lee and Zheng, 2007). The common characteristics of these tools is that (1) they are based on a model that is more general than the one we propose in this paper, and (2) they do not consider composition of discrete timed systems with different periodic clocks. As a consequence, they are not able to provide results as strong as ours when it comes to reasoning about specific global properties of interconnected systems.

7. Concluding remarks

This paper proposes a new mathematical framework for the compositional design of timed heterogeneous systems based on an extension of timed input/output automata (Kaynar et al., 2006; David et al., 2010) where automata are assigned a clock granularity (what we call timed machines). Composition is thus extended to cater for automata that operate over different clock granularities.

One key aspect of our work is that we support the design of heterogeneous timed systems whose clock granularities can be made compatible without modifying the time domains of the individual components. This is important so that components can be interconnected at run time, not design time, which is essential for addressing the new generation of systems that are operating in cyberspace, where they need to be interconnected, on the fly, to other systems. Our approach is truly compositional in that we

can obtain properties of a whole system of interconnected components without having to compute their composition.

Another novel contribution comes from the fact that we distinguish between an algebra of timed machines — where the composition of two timed machines returns another timed machine (their parallel composition) — and an algebra of networks of timed machines — where composition binds networks at run time to create more complex networks. This is important because, in the context of modern systems (of systems) whose structure change as they interconnect, dynamically, with other systems, reducing a system to a (big) component does not make sense: where in traditional software engineering a system is delivered as a whole to a client, networks of systems that operate in cyberspace are not delivered to anyone — they behave autonomously.

The main properties of networks that we address are consistency (there exists at least a joint trace on which all components can agree) and feasibility (there exists at least a joint trace on which all components can agree no matter what input they receive from their environment). The technical results that support compositional verification of consistency and feasibility are based on new notions of time refinement and of cooperation conditions through which timed components can be ensured to be open to interactions with other components across different time granularities.

There are three main directions for future work. The first is to implement and evaluate our approach on concrete case studies. A possibility would be to implement the framework as an extension of Ptolemy (Buck et al., 1994), which would give us access to industrial-size case studies. The second aims at defining an operational semantics that captures the symbiosis that exists between execution and binding, i.e, where state executions within a network trigger the binding to another network, and where binding affects the execution of a network — a preliminary approach has been presented in (Chauchat, 2015). The third is to develop a logic and interface algebra for networks that is similar to the one presented in (Fiadeiro and Lopes, 2013) but adapted to a real-time context and reflecting run-time binding.

Acknowledgments

We would like to thank Paul Chauchat for many important observations and corrections to earlier versions of this paper. We would also like to thank the reviewers for many useful observations and remarks.

This work was partially supported by the Engineering and Physical Sciences Research Council UK (EPSRC) through the grant EP/K000683/1, by the Royal Society International Exchange grant IE130154 and by the AFOSR grant FA9550-14-1-0043 “Semantic Completions: Unifying the Wave and the Particle Views of Information”.

References

Abadi, M. and Lamport, L. (1991). The existence of refinement mappings. *Theor. Comput. Sci.*, 82(2):253–284.

Alur, R. and Henzinger, T. A. (1999). Reactive modules. *Formal Methods in System Design*, 15(1):7–48.

Attie, P. C. and Lynch, N. A. (2001). Dynamic input/output automata: A formal model for dynamic systems. In *CONCUR 2001 - Concurrency Theory, 12th International Conference, Aalborg, Denmark, August 20-25, 2001, Proceedings*, pages 137–151.

Attie, P. C. and Lynch, N. A. (2016). Dynamic input/output automata: A formal and compositional model for dynamic systems. *Information and Computation*.

Basu, A., Bensalem, S., Bozga, M., Combaz, J., Jaber, M., Nguyen, T.-H., and Sifakis, J. (2011). Rigorous component-based system design using the BIP framework. *IEEE Software*, 28(3):41–48.

Benveniste, A., Caillaud, B., Carloni, L. P., and Sangiovanni-Vincentelli, A. L. (2005). Tag machines. In *EMSOFT*, pages 255–263. ACM.

Bliudze, S. and Krob, D. (2009). Modelling of complex systems: Systems as dataflow machines. *Fundam. Inform.*, 91(2):251–274.

Bozga, M., Jaber, M., Maris, N., and Sifakis, J. (2012). Modeling dynamic architectures using dy-BIP. In *Proceedings of the 11th International Conference on Software Composition*, SC'12, pages 1–16, Berlin, Heidelberg. Springer-Verlag.

Broy, M. and Stølen, K. (2001). *Specification and Development of Interactive Systems: Focus on Streams, Interfaces, and Refinement*. Springer-Verlag New York, Inc., Secaucus, NJ, USA.

Buck, J. T., Ha, S., Lee, E. A., and Messerschmitt, D. G. (1994). Ptolemy: A framework for simulating and prototyping heterogenous systems. *Int. Journal in Computer Simulation*, 4(2):155–182.

Burns, A. and Hayes, I. J. (2010). A timeband framework for modelling real-time systems. *Real-Time Syst.*, 45(1-2):106–142.

Chauchat, P. (2015). Asynchronous relational networks. Rapport de M2, Université Paris 7 – Denis-Diderot.

Chen, Y., Chen, Y., and Madelaine, E. (2015). Timed-pNets: a communication behavioural semantic model for distributed systems. *Frontiers of Computer Science*, 9(1):87–110.

David, A., Larsen, K. G., Legay, A., Nyman, U., and Wasowski, A. (2010). Timed I/O automata: a complete specification theory for real-time systems. In *HSCC*, pages 91–100. ACM.

Delahaye, B., Fiadeiro, J. L., Legay, A., and Lopes, A. (2013). A timed component algebra for services. In Beyer, D. and Boreale, M., editors, *FORTE*, volume 7892 of *LNCS*, pages 242–257. Springer.

Delahaye, B., Fiadeiro, J. L., Legay, A., and Lopes, A. (2014). Heterogeneous timed machines. In *ICTAC*, volume 8687 of *LNCS*, pages 115–132. Springer.

Fiadeiro, J. L. and Lopes, A. (2010). A model for dynamic reconfiguration in service-oriented architectures. In Babar, M. A. and Gorton, I., editors, *ECSA*, volume 6285 of *LNCS*, pages 70–85. Springer.

Fiadeiro, J. L. and Lopes, A. (2013). An interface theory for service-oriented design. *Theor. Comput. Sci.*, 503:1–30.

Fiadeiro, J. L. and Lopes, A. (2017). Heterogeneous and asynchronous networks of timed systems. *Theoretical Computer Science*, 663:1–33.

Fiadeiro, J. L., Lopes, A., and Bocchi, L. (2011). An abstract model of service discovery and binding. *Formal Asp. Comput.*, 23(4):433–463.

Fisher, J., Henzinger, T. A., Nickovic, D., Piterman, N., Singh, A. V., and Vardi, M. Y. (2011). Dynamic reactive modules. In Katoen, J. and König, B., editors, *CONCUR 2011*, volume 6901 of *LNCS*, pages 404–418. Springer.

Forget, J., Boniol, F., Lesens, D., and Pagetti, C. (2008). A multi-periodic synchronous data-flow language. In *HASE*, pages 251–260. IEEE Computer Society.

Golden, B., Aiguier, M., and Krob, D. (2012). Modeling of complex systems II: A minimalist and unified semantics for heterogeneous integrated systems. *Applied Mathematics and Computation*, 218(16):8039–8055.

Gößler, G. and Sangiovanni-Vincentelli, A. L. (2002). Compositional modeling in metropolis. In *EMSOFT*, volume 2491 of *LNCS*, pages 93–107. Springer.

Grötker, T. (2002). *System Design with SystemC*. Springer.

Henzinger, T. A., Manna, Z., and Pnueli, A. (1991). Timed transition systems. In de Bakker, J. W., Huizing, C., de Roever, W. P., and Rozenberg, G., editors, *REX Workshop*, volume 600 of *LNCS*, pages 226–251. Springer.

Kaynar, D. K., Lynch, N., Segala, R., and Vaandrager, F. (2006). *The Theory of Timed I/O Automata*. Morgan & Claypool Publishers.

Le, T., Passerone, R., Fahrenberg, U., and Legay, A. (2013). A tag contract framework for heterogeneous systems. In Canal, C. and Villari, M., editors, *ESOCC Workshops*, volume 393 of *CCIS*, pages 204–217. Springer.

Lee, E. A. and Zheng, H. (2007). Leveraging synchronous language principles for heterogeneous modeling and design of embedded systems. In *EMSOFT*, pages 114–123. ACM.

Papazoglou, M. P., Traverso, P., Dustdar, S., and Leymann, F. (2007). Service-oriented computing: State of the art and research challenges. *IEEE Computer*, 40(11):38–45.

Sander, I. and Jantsch, A. (2004). System modeling and transformational design refinement in ForSyDe [formal system design]. *IEEE Trans. on CAD of Integrated Circuits and Systems*, 23(1):17–32.

Appendix

Proof of Lemma 3.1

For all locations l and clock valuations v and v' such that $v \sim v'$, the states (l, v) and (l, v') of $\mathcal{M}$ have equivalent outgoing transitions, i.e. whenever state (l, v) can take a transition $(l, S, C, R, l') \in E$ and end up in (l', v''), state (l, v') can take the same transition and end up in (l', v''') with $v'' \sim v'''$, and vice versa.

Proof. Let $l \in Loc$ be a location and $v \sim v'$ be two equivalent clock valuations. Assume that there exists a transition $(l, S, C, R, l') \in E$ enabled in (l, v) and ending up in (l', v''). By construction, we thus have

—for all $0 \leq t \leq \delta$, $v + t \models Inv(l)$,
—$v'' = (v + \delta)^{\mathbf{R}}$, and
—$v + \delta \models C$.

We now verify that the same holds for (l, v'). Consider a clock $c \in \mathbb{C}$. Because $v \sim v'$, we have that either (a) $v(c) = v'(c)$ or (b) $v(c) > \mathsf{Max}^{\mathcal{A}}(c)$ and $v'(c) > \mathsf{Max}^{\mathcal{A}}(c)$. In both cases, we obtain that (l, S, C, R, l') is enabled in (l, v') and ends up in (l', v''') with $v'' \sim v'''$. □

Proof of Theo. 3.1

For all action sequences σ over Act, $\langle \sigma, \tau_{\delta_{\mathcal{M}}} \rangle$ is a timed trace defined by an execution in $[\![\mathcal{M}]\!]$ iff $\sigma(1)\sigma(2)\ldots \in \mathcal{L}(\mathcal{B}_{\mathcal{M}})$, where $\mathcal{L}$ denotes the language of the Büchi automaton.

Proof. Let $\lambda = \langle \sigma, \tau_\delta \rangle$ be a timed trace over Act and assume that it is defined by the execution in $[\![\mathcal{M}]\!]$

$$(l_0, v_0) \xrightarrow{\sigma(1), R_1} (l_1, v_1) \xrightarrow{\sigma(2), R_2} (l_2, v_2) \cdots$$

where (l_0, v_0) is necessarily $(l^\star, v^\star)$.

By definition of $\mathcal{M}$, all involved valuations v_i are such that $v_i(c)$ is a multiple of δ for all c. As a consequence, we define $\boldsymbol{\nu}_i$ to be such that, for all $c \in \mathbb{C}$, $\boldsymbol{\nu}_i(c) = v_i(c)/\delta$ if $v_i(c) \leq \mathsf{Max}^{\mathcal{A}}(c)$ and $\boldsymbol{\nu}_i(c) = \lfloor \frac{\mathsf{Max}^{\mathcal{A}}(c)}{\delta} \rfloor + 1$ otherwise. Notice that, in this way, $(l^\star, \boldsymbol{\nu}_0)$ coincides with the initial state of the Büchi automaton.

Consider now a state (l_i, v_i) from the above execution. Because there exists a transition

$$(l_i, \sigma(i+1), C_{i+1}, R_{i+1}, l_{i+1}) \in E$$

that is enabled in (l_i, v_i), there is a corresponding transition $(l_i, \boldsymbol{\nu}_i) \xrightarrow{\sigma(i+1)} (l_{i+1}, \boldsymbol{\nu}_{i+1})$ in $\mathcal{B}_{\mathcal{M}}$. As a consequence, we obtain that $\sigma(1)\sigma(2)\ldots \in \mathcal{L}(\mathcal{B}_{\mathcal{M}})$.

Reversely, assume that $\sigma(1)\sigma(2)\ldots \in \mathcal{L}(\mathcal{B}_{\mathcal{M}})$. By construction, there exists a corresponding execution in $\mathcal{B}_{\mathcal{M}}$ as follows:

$$q_0 \xrightarrow{\sigma(1)} (l_1, \boldsymbol{\nu}_1) \xrightarrow{\sigma(2)} \cdots$$

We have that $q_0 = (l^\star, v^\star/\delta)$ where $v^\star/\delta$ is the clock valuation that, for every $c \in \mathbb{C}$, assigns $v^\star(c)/\delta$ if $v^\star(c) \leq \mathsf{Max}^{\mathcal{A}}(c)$ and assigns $\lfloor \frac{\mathsf{Max}^{\mathcal{A}}(c)}{\delta} \rfloor + 1$ otherwise.

We build the corresponding execution from the initial state $(l^\star, v^\star)$ of $\mathcal{M}$. From the initial state $(l^\star, v^\star/\delta)$ of the Büchi automaton, there is a transition $(l^\star, v^\star/\delta) \xrightarrow{\sigma(1)} (l_1, \boldsymbol{\nu}_1)$ in $\mathcal{B}_\mathcal{M}$. Therefore, there exists a corresponding transition $(l^\star, \sigma(1), C_1, R_1, l_1)$ in $\mathcal{M}$ such that $(l^\star, v^\star) \xrightarrow{\sigma(1), R_1} (l_1, v_1)$ in $\mathcal{M}$, with $v_1 = (v^\star + \delta)^{\mathbf{R_1}}$. In addition, we can show that $v_1(c) = \boldsymbol{\nu}_1(c) \cdot \delta$ for all $c \in \mathbb{C}$ such that $v_1(c) \leq \mathsf{Max}^\mathcal{A}(c) + \delta$ and $\boldsymbol{\nu}_1(c) = \lfloor \frac{\mathsf{Max}^\mathcal{A}(c)}{\delta} \rfloor + 1$ otherwise. By Lemma 3.1, all transitions that can be taken from (l_1, v_1) can also be taken from $(l_1, \boldsymbol{\nu}_1 \cdot \delta)$ and vice versa. Thus, because there exists a transition

$$(l_1, \boldsymbol{\nu}_1) \xrightarrow{\sigma(2)} (l_2, \boldsymbol{\nu}_2)$$

in $\mathcal{B}_\mathcal{M}$, there is a corresponding transition from $(l_1, \boldsymbol{\nu}_1 \cdot \delta)$ and hence the same holds for (l_1, v_1) in $\mathcal{M}$.

Following this reasoning, we build an execution of $\mathcal{M}$ as follows:

$$(l^\star, v^\star) \xrightarrow{\sigma(1), R_1} (l_1, v_1) \xrightarrow{\sigma(2), R_2} (l_2, v_2) \cdots$$

where for all v_i, either $v_i(c) = \boldsymbol{\nu}_i(c) \cdot \delta$ or $v_i(c) > \mathsf{Max}^\mathcal{A}(c) + \delta$ and $\boldsymbol{\nu}_i(c) = \lfloor \frac{\mathsf{Max}^\mathcal{A}(c)}{\delta} \rfloor + 1$. □

Proof of Prop. 4.2

Any initializable DTIOM that makes independent progress is consistent.

Proof. Let $\mathcal{M}$ be a DTIOM that is initializable and makes independent progress. We construct an execution of $\mathcal{M}$

$$(l^\star, v^\star) = (l_0, v_0) \xrightarrow{S_0, R_0} (l_1, v_1) \xrightarrow{S_1, R_1} (l_2, v_2) \xrightarrow{S_2, R_2} \ldots$$

by choosing each edge $(l_i, S_i, C_i, R_i, l_{i+1})$ as guaranteed by the definition of making independent progress and $v_{i+1} = (v_i + \delta_\mathcal{M})^{\mathbf{R_i}}$. The construction is good because $\mathcal{M}$ initializable guarantees that $(l^\star, v^\star)$ is reachable and the definition of making independent progress applied to (l_i, v_i) guarantees that (l_{i+1}, v_{i+1}) is reachable. □

Proof of Lem. 4.1

Let $k \in \mathbb{N}_{>0}$.

(a) If a DTIOM $\mathcal{M}$ is initializable then so is $\mathcal{M}_k$.
(b) If a DTIOM $\mathcal{M}$ can make independent progress then so can $\mathcal{M}_k$.
(c) If a DTIOM $\mathcal{M}$ is DP-enabled in relation to $J \subseteq Act^I_\mathcal{M}$ and a multiple δ of $\delta_\mathcal{M}$ then $\mathcal{M}_k$ is DP-enabled in relation to J and δ.
(d) Let $\mathcal{M}$ be a cooperative DTIOM with respect to $Q \subseteq Act_\mathcal{M}$ and a multiple δ of $\delta_\mathcal{M}$. Then $\mathcal{M}_k$ is also cooperative in relation to Q and δ'.

Proof. Let $k \in \mathbb{N}_{>0}$.

(a) The result follows trivially from the construction of $\mathcal{M}_k$: the initial location of $\mathcal{M}_k$ is $(l^\star, 0)$ and its invariant is $Inv_\mathcal{M}(l^\star)$, and the initial clock valuation of $\mathcal{M}_k$ is the same

as that of $\mathcal{M}$ and, hence, if $v^\star+t \vDash Inv(l^\star)$, for $0 \leq t \leq \delta_M$, then $v^\star+t \vDash Inv_{\mathcal{M}_k}(l^\star, 0)$, for $0 \leq t \leq \delta_M/k$.

(b) Let $\mathcal{M}$ be a DTIOM that can make independent progress and $\delta = \delta_{\mathcal{M}}$. Consider $(l, i) \in Loc_{\mathcal{M}_k}$ such that $((l, i), v)$ is reachable in $\mathcal{M}_k$. We want to prove that $\mathcal{M}_k$ can make independent progress from that state.

By the construction of $\mathcal{M}_k$ (cf. Def. 3.4), we know that if $((l, i), v)$ is reachable at a time T in $\mathcal{M}_k$ then $T = T' + i.\delta/k$ for some time T' such that $(l, v' = v - i.\delta/k)$ is reachable in $\mathcal{M}$ at time T' (i.e., we wind up the clock to the previous 'tick' of $\mathcal{M}$ to find a state of $\mathcal{M}$ with the same location that is necessarily reachable).

We have two cases to consider depending on whether T corresponds to the last 'tick' of $\mathcal{M}_k$ before $T' + \delta_{\mathcal{M}}$ (in which case $\mathcal{M}_k$ performs the corresponding transition of $\mathcal{M}$), or T corresponds to an intermediate point between T' and $T' + \delta_{\mathcal{M}}$:

—If $i = k - 1$, because $\mathcal{M}$ can make independent progress and $(l, v' = v - i.\delta/k)$ is reachable in $\mathcal{M}$, we know that there exists an edge $(l, A, C, R, l') \in E_{\mathcal{M}}$ such that: (a) $A \subseteq Act^O \cup Act^\tau$, (b) $v'+\delta \vDash C$ and (c) for all $0 \leq t \leq \delta$, $(v'+\delta)^{\mathbf{R}}+t \vDash Inv(l')$. By construction, $((l, i), A, C, R, (l', 0)) \in E_{\mathcal{M}_k}$. Because

$$v + \delta/k = v' + (k - 1).\delta/k + \delta/k = v' + \delta$$

we conclude that (a) $v + \delta/k \vDash C$ and (b) for all $0 \leq t \leq \delta/k$, $(v + \delta/k)^{\mathbf{R}} + t \vDash Inv_k((l', 0))$. Therefore, $\mathcal{M}_k$ can make independent progress when at $((l, i), v)$ with $i = k - 1$.

—If $i \in [0..k - 2]$ then, by construction, $((l, i), \emptyset, \mathsf{true}, \emptyset, (l, i + 1)) \in E_{\mathcal{M}_k}$ and $Inv_{\mathcal{M}_k}((l, i+1)) = Inv_{\mathcal{M}}(l)$. Moreover, because (l, v') is reachable in $\mathcal{M}$, we have that $v' + t \vDash Inv_{\mathcal{M}}(l)$ for all $0 \leq t \leq \delta$. Because $v = v' + i.\delta/k$, we can conclude that $(v + (\delta/k)) + t \vDash Inv_{\mathcal{M}_k}((l, i + 1))$, for all $0 \leq t \leq \delta/k$. Therefore, $\mathcal{M}_k$ can make independent progress when at $((l, i), v)$ with $i \in [0..k - 2]$.

(c) Let $\mathcal{M}$ be a DP-enabled DTIOM in relation to $J \subseteq Act^I_{\mathcal{M}}$.

Consider $B \subseteq J$, an edge $((l, i), A, C, R, (l', i')) \in E_{\mathcal{M}_k}$ and a clock valuation v such that: (a) $((l, i), v)$ is reachable in $\mathcal{M}_k$ at a time T such that $T + \delta_{\mathcal{M}}/k$ is a multiple of δ, (b) $v + \delta_{\mathcal{M}}/k \vDash C$, and (c) for all $0 \leq t \leq \delta_{\mathcal{M}}/k$, $(v + \delta_{\mathcal{M}}/k)^{\mathbf{R}} + t \vDash Inv((l', i'))$.

By construction, we know that if $((l, i), v)$ is reachable at a time T in $\mathcal{M}_k$ then $T = T' + i.\delta_{\mathcal{M}}/k$ for some time T' such that $(l, v' = v - i.\delta_{\mathcal{M}}/k)$ is reachable in $\mathcal{M}$ at time T'. Because $T + \delta_{\mathcal{M}}/k$ is a multiple of δ (and, hence, of $\delta_{\mathcal{M}}$), we have that $i = k - 1$. Again by construction, we have that $i' = 0$, $(l, A, C, R, l') \in E_{\mathcal{M}}$ and $v + \delta_{\mathcal{M}}/k = v' + \delta_{\mathcal{M}}$.

Because $\mathcal{M}$ is DP-enabled, there exists an edge $(l, B \cup (A \setminus J), C', R', l'') \in E_{\mathcal{M}}$ such that (a) $v' + \delta_{\mathcal{M}} \vDash C'$, and (b) for all $0 \leq t \leq \delta_{\mathcal{M}}$, $(v' + \delta_{\mathcal{M}})^{\mathbf{R}'} + t \vDash Inv(l'')$. By construction, we have that $((l, i), B \cup (A \setminus J), C', R', (l'', 0)) \in E_{\mathcal{M}_k}$. From the properties above we can conclude that (a) $v + \delta_{\mathcal{M}}/k \vDash C'$ and (b) for all $0 \leq t \leq \delta_{\mathcal{M}}/k$, $(v + \delta_{\mathcal{M}}/k)^{\mathbf{R}'} + t \vDash Inv_k((l'', 0))$.

(d) Let $\mathcal{M}$ be a cooperative DTIOM with respect to $Q \subseteq Act_{\mathcal{M}}$ and a multiple δ of $\delta_{\mathcal{M}}$.

Consider an edge $((l,i),A,C,R,(l',i')) \in E_{\mathcal{M}_k}$ and a clock valuation v such that: (a) $((l,i),v)$ is reachable in $\mathcal{M}_k$ at a time T for which $T+\delta_{\mathcal{M}}/k$ is not a multiple of δ (b) $v+\delta_{\mathcal{M}}/k \vDash C$ and (c) for all $0 \leq t \leq \delta_{\mathcal{M}}/k$, $(v+\delta_{\mathcal{M}}/k)^{\mathbf{R}}+t \vDash Inv((l',i'))$. By construction, we know that if $((l,i),v)$ is reachable at a time T in $\mathcal{M}_k$ then $T = T' + i.\delta_{\mathcal{M}}/k$ for some time T' such that $(l, v' = v - i.\delta_{\mathcal{M}}/k)$ is reachable in $\mathcal{M}$ at time T'. We have two cases to consider:

—If $i = k-1$, we have that $i' = 0$, $(l,A,C,R,l') \in E_{\mathcal{M}}$, and $T+\delta_{\mathcal{M}}/k = T'+\delta_{\mathcal{M}}$ (which implies that $T'+\delta_{\mathcal{M}}$ is not a multiple of δ). Because $\mathcal{M}$ is cooperative with respect to Q and δ, we know that there exists an edge $(l, A \setminus Q, C', R', l'') \in E_{\mathcal{M}}$ such that (a) $v'+\delta_{\mathcal{M}} \vDash C'$ and (b) for all $0 \leq t \leq \delta_{\mathcal{M}}$, $(v'+\delta_{\mathcal{M}})^{\mathbf{R}'}+t \vDash Inv_{\mathcal{M}}(l'')$. By construction, it follows that $((l,i), A \setminus Q, C', R', (l'',0)) \in E_{\mathcal{M}_k}$. From the properties above we can also conclude that (a) $v+\delta_{\mathcal{M}}/k \vDash C'$ and (b) for all $0 \leq t \leq \delta_{\mathcal{M}}/k$, $(v+\delta_{\mathcal{M}}/k)^{\mathbf{R}'}+t \vDash Inv_{\mathcal{M}_k}((l'',0))$.

—If $i \in [0..k-2]$, we have, by construction, that $l' = l$, $i' = i+1$, $A = \emptyset$, $C = \mathsf{true}$, and $R = \emptyset$. In this case, the result follows trivially because $A \setminus Q = A$.

□

Preservation of properties under homogeneous composition

For the proof of Lem. 4.2 it is useful to establish first an auxiliary result for homogenous composition.

Lemma 7.1. Let $\mathcal{M}_i = \langle \delta, \mathcal{A}_i \rangle$, $i = 1,2$, be two DTIOMs such that $\mathcal{A}_1$ and $\mathcal{A}_2$ are compatible. Let $\mathcal{M} = \|_{i=1,2}^{(l_i,v_i,t_i)} \mathcal{M}_i$ where, for $i = 1,2$, (l_i, v_i) is a reachable state at time t_i in $\mathcal{M}_i$. Let δ' be a multiple of δ such that t_1 is a multiple of δ'.

(a) If $\mathcal{M}_1$ is DP-enabled in relation to $J \subseteq Act_1^I$ and δ', then $\mathcal{M}$ is DP-enabled in relation to $J \setminus Act_2^O$ and δ'.

(b) If $\mathcal{M}_1$ is cooperative in relation to $Q \subseteq Act_1^O \setminus Act_2^I$ and δ', then $\mathcal{M}$ is cooperative in relation to Q and δ'.

Proof.

(a) Let $\mathcal{M}_1$ be DP-enabled in relation to $J \subseteq Act_1^I$ and δ' a multiple of δ and t_1 a multiple of δ'. Consider $B \subseteq J \setminus Act_2^O$, an edge $((l_1,l_2),A,C_1 \wedge C_2, R_1 \cup R_2, (l_1',l_2')) \in E_{\mathcal{A}_1 \| \mathcal{A}_2}$ and a clock valuation $v_1 \cup v_2$ such that: (a) $((l_1,l_2),v_1 \cup v_2)$ is reachable in $\mathcal{M}$ at a time T such that $T+\delta$ is a multiple of δ', (b) $(v_1 \cup v_2)+\delta \vDash C_1 \wedge C_2$, and (c) for all $0 \leq t \leq \delta$, $((v_1 \cup v_2)+\delta)^{\mathbf{R_1} \cup \mathbf{R_2}} + t \vDash Inv_1(l_1') \wedge Inv_2(l_2')$.

By construction of $\mathcal{M}$ (cf. Def. 2.7), we know that, for $i = 1,2$, there exists an edge $(l_i, A_i, C_i, R_i, l_i') \in E_i$ such that: (a) $A_i = A \cap Act_i$, (b) $v_i + \delta \vDash C_i$, and (c) for all $0 \leq t \leq \delta$, $(v_i+\delta)^{\mathbf{R_i}} + t \vDash Inv_i(l_i')$. By the construction of $\mathcal{M}$, we also know that (l_1, v_1) is reachable in $\mathcal{A}_1$ at a time $t_1 + T$. The fact that t_1 is a multiple of δ' together with the hypothesis that $T+\delta$ is a multiple of δ' implies that $t_1 + T + \delta$ is a multiple of δ',

Because $\mathcal{M}_1$ is DP-enabled in relation to J and δ', there exists an edge $(l_1, B \cup (A_1 \setminus J)), C_1', R_1', l_1'') \in E_1$ such that: (a) $v_1 + \delta \vDash C_1'$, and (b) for all $0 \leq t \leq \delta$, $(v_1 + \delta)^{\mathbf{R_1'}} + t \vDash Inv_1(l_1'')$.

Let $A' = B \cup (A \setminus J)$. On the one hand, because $B \subseteq J \setminus Act_2^O \subseteq Act_1^I$ and $A_1 = A \cap Act_1$, we have that $A' \cap Act_1 = (B \cap Act_1) \cup ((A \setminus J) \cap Act_1) = B \cup (A_1 \setminus J)$. On the other hand, because $A_2 = A \cap Act_2$, we have that $A' \cap Act_2 = (B \cap Act_2) \cup ((A \setminus J) \cap Act_2) = (A \setminus J) \cap Act_2 = A_2$.

Then, by construction, there exists an edge $((l_1, l_2), B \cup (A \setminus J), C_1 \wedge C_2, R_1 \cup R_2, (l_1'', l_2')) \in E_{\mathcal{A}_1 \| \mathcal{A}_2}$ such that: (a) $(v_1 \cup v_2) + \delta \vDash C_1' \wedge C_2$, and (b) for all $0 \leq t \leq \delta$, $((v_1 \cup v_2) + \delta)^{\mathbf{R_1' \cup R_2}} + t \vDash Inv_1(l_1'') \wedge Inv_2(l_2')$. Hence, we can conclude that $\mathcal{M}$ is DP-enabled in relation to $J \setminus Act_2^O$ and δ'.

(b) Let $\mathcal{M}_1$ be cooperative in relation to $Q \subseteq Act_1^O \setminus Act_2^I$ and δ', a multiple of δ such that t_1 is a multiple of δ'. Consider an edge $((l_1, l_2), A, C_1 \wedge C_2, R_1 \cup R_2, (l_1', l_2')) \in E_{\mathcal{A}_1 \| \mathcal{A}_2}$ and a clock valuation $v_1 \cup v_2$ such that: (a) $((l_1, l_2), v_1 \cup v_2)$ is reachable in $\mathcal{M}$ at a time T such that $T + \delta$ is not a multiple of δ', (b) $(v_1 \cup v_2) + \delta \vDash C_1 \wedge C_2$, and (c) for all $0 \leq t \leq \delta$, $((v_1 \cup v_2) + \delta)^{\mathbf{R_1 \cup R_2}} + t \vDash Inv_1(l_1') \wedge Inv_2(l_2')$.

By the construction of $\mathcal{A}_1 \| \mathcal{A}_2$ (cf. Def. 2.7), we know that, for $i = 1, 2$, there exists an edge $(l_i, A_i, C_i, R_i, l_i') \in E_i$ such that: (a) $A_i = A \cap Act_i$, (b) $v_i + \delta \vDash C_i$, and (c) for all $0 \leq t \leq \delta$, $(v_i + \delta)^{\mathbf{R_i}} + t \vDash Inv_i(l_i')$. By the construction of $\mathcal{M}$, we also know that (l_1, v_1) is reachable in $\mathcal{A}_1$ at a time $t_1 + T$. The fact that t_1 is a multiple of δ' together with the hypothesis that $T + \delta$ is not a multiple of δ' implies that $t_1 + T + \delta$ is not a multiple of δ',

Because $\mathcal{M}_1$ is cooperative in relation to Q and δ', there exists an edge $(l_1, A_1 \setminus Q, C_1', R_1', l_1'') \in E_1$ such that: (a) $v_1 + \delta \vDash C_1'$, and (b) for all $0 \leq t \leq \delta$, $(v_1 + \delta)^{\mathbf{R_1'}} + t \vDash Inv_1(l_1'')$.

Let $A' = A \setminus Q$. On the one hand, because $Q \subseteq Act_1^O \setminus Act_2^I$, $A' \cap Act_1 = (A \cap Act_1) \setminus Q = A_1 \setminus Q = A_1'$. On the other hand, we have that $Q \cap Act_2 = \emptyset$ and, hence, $A' \cap Act_2 = A \cap Act_2 = A_2$.

Then, by construction, there exists an edge $((l_1, l_2), A \setminus Q, C_1 \wedge C_2, R_1' \cup R_2, (l_1'', l_2')) \in E_{\mathcal{A}_1 \| \mathcal{A}_2}$ such that: (a) $(v_1 \cup v_2) + \delta \vDash C_1' \wedge C_2$, and (b) for all $0 \leq t \leq \delta$, $((v_1 \cup v_2) + \delta)^{\mathbf{R_1' \cup R_2}} + t \vDash Inv_1(l_1'') \wedge Inv_2(l_2')$. Hence, we can conclude that $\mathcal{M}$ is cooperative in relation to Q and δ'.

□

Proof of Lem. 4.2

Let $\mathcal{M}_i = \langle \delta_i, \mathcal{A}_i \rangle$, $i = 1, 2$, be two δ-compatible DTIOMs. Let $\mathcal{M} = \mathcal{M}_i \Big|\Big|_{\delta}^{(l_i, v_i, t_i)}$ where, for $i = 1, 2$, (l_i, v_i) is a reachable state at time t_i in $\mathcal{M}_i$. Let δ_1' be a multiple of δ_1 such that t_1 is a multiple of δ_1'.

(a) If $\mathcal{M}_1$ is DP-enabled in relation to $J \subseteq Act_1^I$ and δ_1', then $\mathcal{M}$ is DP-enabled in relation to $J \setminus Act_2^O$ and δ_1'.
(b) If $\mathcal{M}_1$ is cooperative in relation to $Q \subseteq Act_1^O \setminus Act_2^I$ and δ_1', then $\mathcal{M}$ is cooperative in relation to Q and δ_1'.

Proof.

—Let $k_i = \delta_i/\delta$. Lemma 4.1 (c) allows us to infer that $\mathcal{M}_{ik_i}$ is DP-enabled in relation to J and δ_1'. Because δ_1' is a multiple of δ_1 and, hence of δ, we are in conditions of applying Lem. 7.1 (a), which ensures that $\mathcal{M}$ is DP-enabled in relation to $J \setminus Act_2^O$ and δ_1'.

—Let $k_i = \delta_i/\delta$. Lem. 4.1 (d) allows us to infer that $\mathcal{M}_{ik_i}$ is cooperative in relation to Q and δ_1'. Because δ_1' is a multiple of δ_1 and, hence of δ, we are in conditions of applying Lem. 7.1 (b), which ensures that $\mathcal{M}$ is cooperative in relation to Q and δ_1'.

□

Preservation of independent progress under homogeneous composition

For the proof of Theo. 4.1 it is useful to establish first an auxiliary result for homogenous composition.

Lemma 7.2. Let $\mathcal{M}_i = \langle \delta, \mathcal{A}_i \rangle$, $i = 1, 2$, be two DTIOMs that can make independent progress such that $\mathcal{A}_1$ and $\mathcal{A}_2$ are compatible. Let

$$\mathcal{M} = \bigparallel_{i=1,2}^{(l_i, v_i, t_i)} \mathcal{M}_i$$

where, for $i = 1, 2$, (l_i, v_i) is a reachable state at time t_i in $\mathcal{M}_i$.

$\mathcal{M}$ is initializable and, if, for some δ' multiple of δ such that t_1 and t_2 are multiples of δ', $\mathcal{M}_1$ is DP-enabled in relation to $Act_1^I \cap Act_2^O$ and δ', $\mathcal{M}_2$ is DP-enabled in relation to $Act_2^I \cap Act_1^O$ and δ', and both $\mathcal{M}_1$ and $\mathcal{M}_2$ are δ'-cooperative in relation to $Act_1 \cap Act_2$, then $\mathcal{M}$ makes independent progress (and, hence, by Prop. 4.2, is consistent).

Proof. Let $\mathcal{A}_i = \langle Loc^i, l^{\star i}, \mathbb{C}^i, v^{\star i}, E^i, Act_i, Inv^i \rangle$, for $i = 1, 2$.

The fact that $\mathcal{M}$ is initializable follows from (1) the fact that, by definition of $\mathcal{M}$, its initial state is $((l_1, l_2), v_1 \cup v_2)$ and $Inv(l_1, l_2) = Inv_1(l_1) \wedge Inv_2(l_2)$, and (2) the fact that each (l_i, v_i) is a reachable state in $\mathcal{M}_i$ (and, hence, for all $0 \leq t \leq \delta$, $v_i + t \vDash Inv_i(l_i)$).

Consider state $((l_1', l_2'), v_1' \cup v_2')$ reachable in $\mathcal{M}$ at time T. By definition of $\mathcal{M}$ and because, each (l_i, v_i) is a reachable state at time t_i in $\mathcal{M}_i$, it follows that, for $i = 1, 2$, (l_i', v_i') is reachable in $\mathcal{M}_i$ at time $t_i + T$. Because each $\mathcal{M}_i$ can make independent progress, for $i = 1, 2$, there is an edge $(l_i', A_i, C_i, R_i, l_i'') \in E_i$ such that (a) $A_i \subseteq Act_i^O \cap Act_i^\tau$, (b) $v_i' + \delta \vDash C_i$, and (c) for all $0 \leq t \leq \delta$, $(v_i' + \delta)^{\mathbf{R_i}} + t \vDash Inv_i(l_i'')$. We have two cases to consider:

—T is multiple of δ':

In this case, because each t_i is a multiple of δ', so is $t_i + T$. Let $A' = (A_1 \setminus (Act_1^I \cap Act_2^O)) \cup (A_2 \setminus (Act_2^I \cap Act_1^O))$ and $A'_i = A' \cap Act_i$.

We have that $A'_1 = (A_1 \setminus (Act_1^I \cap Act_2^O)) \cup (A_2 \cap Act_1^I)$. Because $\mathcal{M}_1$ is DP-enabled in relation to $Act_1^I \cap Act_2^O$ and δ', $t_i + T$ is multiple of δ' and $A_2 \cap Act_1^I \subseteq Act_1^I \cap Act_2^O$, there is an edge $(l'_1, A'_1, C'_1, R'_1, l'''_1) \in E_1$ such that (a) $v'_1 + \delta \vDash C'_1$, and (b) for all $0 \leq t \leq \delta$, $(v'_1 + \delta)^{\mathbf{R'_1}} + t \vDash Inv_1(l'''_1)$. Similarly, we can conclude that there is an edge $(l'_2, A'_2, C'_2, R'_2, l'''_2) \in E_2$ such that (a) $v'_2 + \delta \vDash C'_2$, and (b) for all $0 \leq t \leq \delta$, $(v'_2 + \delta)^{\mathbf{R'_2}} + t \vDash Inv_1(l'''_2)$.

Then, by construction of $\mathcal{A}_1 \parallel \mathcal{A}_2$ and $\mathcal{M}$, there exists an edge $((l'_1, l'_2), A', C'_1 \wedge C'_2, R'_1 \cup R'_2, (l'''_1, l'''_2))$ in $\mathcal{M}$ such that: (a) $(v'_1 \cup v'_2) + \delta \vDash C'_1 \wedge C'_2$, and (b) for all $0 \leq t \leq \delta$, $((v'_1 \cup v'_2) + \delta)^{\mathbf{R'_1} \cup \mathbf{R'_2}} + t \vDash Inv_1(l'''_1) \wedge Inv_2(l'''_2)$.

—T is not a multiple of δ':

In this case, because each t_i is a multiple of δ', $t_i + T$ is also not a multiple of δ'. Let $A' = A_1 \setminus (Act_1 \cap Act_2) \cup A_2 \setminus (Act_1 \cap Act_2)$ and $A'_i = A_i \cap Act_i$.

We have that, for $i = 1, 2$, $A'_i = A_1 \setminus (Act_1 \cap Act_2)$. Because, for $i = 1, 2$, $\mathcal{M}_i$ is δ'-cooperative in relation to $Act_1 \cap Act_2$, we have that there is an edge $(l'_i, A'_i, C'_i, R'_i, l'''_i) \in E_i$ such that (a) $v'_i + \delta \vDash C'_i$, and (b) for all $0 \leq t \leq \delta$, $(v'_i + \delta)^{\mathbf{R'_i}} + t \vDash Inv_i(l'''_i)$.

Then, by construction of $\mathcal{A}_1 \parallel \mathcal{A}_2$ and $\mathcal{M}$, there exists an edge $((l'_1, l'_2), A', C'_1 \wedge C'_2, R'_1 \cup R'_2, (l'''_1, l'''_2))$ in $\mathcal{M}$ such that: (a) $(v'_1 \cup v'_2) + \delta \vDash C'_1 \wedge C'_2$, and (b) for all $0 \leq t \leq \delta$, $((v'_1 \cup v'_2) + \delta)^{\mathbf{R'_1} \cup \mathbf{R'_2}} + t \vDash Inv_1(l'_1) \wedge Inv_2(l''_2)$.

□

Proof of Theo. 4.1

Let $\mathcal{M}_i$, $i = 1, 2$, be two δ-compatible DTIOMs that can make independent progress. Let

$$\mathcal{M} = {}_{\delta}\Big|\Big|_{i=1,2}^{(l_i, v_i, t_i)} \mathcal{M}_i$$

where, for $i = 1, 2$, (l_i, v_i) is a reachable state at time t_i in $\mathcal{M}_i$.

If, for some δ' multiple of δ_1 and δ_2 such that t_1 and t_2 are multiples of δ', $\mathcal{M}_1$ is DP-enabled in relation to $Act_1^I \cap Act_2^O$ and δ', $\mathcal{M}_2$ is DP-enabled in relation to $Act_2^I \cap Act_1^O$ and δ', and both $\mathcal{M}_1$ and $\mathcal{M}_2$ are δ'-cooperative in relation to $Act_1 \cap Act_2$, then $\mathcal{M}$ is initializable and makes independent progress (and, hence, by Prop. 4.2, is consistent).

Proof. From Lem. 7.2 follows that $\mathcal{M}$ is initializable. Let $k_i = \delta_i / \delta$. Lem. 4.1 allows us to infer that $\mathcal{M}_{ik_i}$ makes independent progress, is DP-enabled in relation to $Act_1^I \cap Act_2^O$ and δ', and is cooperative in relation to $Act_1 \cap Act_2$ and δ'. Because δ' is a multiple of δ, we can use Prop. 7.2, which ensures that $\mathcal{M}$ can make independent progress. □

Proof of Prop. 4.3

A DTIOM $\mathcal{M}$ that is initializable, makes independent progress and is DP-enabled in relation to $J \subseteq Act_{\mathcal{M}}^I$ and a multiple δ of $\delta_{\mathcal{M}}$ is feasible in relation to J and δ.

Proof. Let $\mathcal{M}$ be a DTIOM that is initializable, makes independent progress and is DP-enabled in relation to $J \subseteq Act^I_{\mathcal{M}}$ and a multiple δ of $\delta_{\mathcal{M}}$. Let $\lambda = \langle \sigma, \tau_\delta \rangle$ be a δ-timed trace over J and (l_0, v_0) a reachable state at a time T_0 such that $T_0 + \delta_{\mathcal{M}}$ is a multiple of δ. We construct an execution of $\mathcal{M}$ starting in (l_0, v_0)

$$(l_0, v_0) \stackrel{S_0,R_0}{\longrightarrow} (l_1, v_1) \stackrel{S_1,R_1}{\longrightarrow} (l_2, v_2) \stackrel{S_2,R_2}{\longrightarrow} \dots$$

as follows. At each state (l_i, v_i), we start by choosing (l_i, S, C, R, l') as guaranteed by the definition of making independent progress. Notice that, by construction, $S \cap Act^I_{\mathcal{M}} = \emptyset$.

Let suppose that $T_0 = n.\delta_{\mathcal{M}}$. If $(n+1).\delta_{\mathcal{M}}$ is not a multiple of δ, then we keep that edge: (a) $l_{i+1} = l'$, (b) $S_i = S$, (c) $R_i = R'$, and (d) $v_{i+1} = (v_i + \delta_{\mathcal{M}})^{\mathbf{R}'}$.

If $(n+1).\delta_{\mathcal{M}}$ is a multiple of δ, then we use the property of being DP-enabled to add to S the inputs of $\sigma(i+1)$: we choose $(l_i, \sigma((i+1)/\delta_{\mathcal{M}}) \cup S, C', R', l'')$ and define: (a) $l_{i+1} = l''$, (b) $S_i = \sigma((i+1)/\delta_{\mathcal{M}}) \cup S$, (c) $R_i = R'$, and (d) $v_{i+1} = (v_i + \delta_{\mathcal{M}})^{\mathbf{R}'}$.

Notice that the state (l_{i+1}, v_{i+1}) is again reachable, ensuring that the construction can be iterated. It is also evident that the $\delta_{\mathcal{M}}$-timed trace obtained from this execution is a refinement of λ. □

Proof of Theo. 4.2

Let $\mathcal{M}_i$, $i = 1, 2$, be two δ-compatible DTIOMs that can make independent progress. Let

$$\mathcal{M} = {}_{\delta}\Big|\Big|_{i=1,2}^{(l_i, v_i, t_i)} \mathcal{M}_i$$

where, for $i = 1, 2$, (l_i, v_i) is a reachable state at time t_i in $\mathcal{M}_i$. Let δ'_1 be a multiple of δ_1 and $J \subseteq Act^I_1$. If

(1) $\mathcal{M}_1$ is DP-enabled in relation to J and δ'_1, and
(2) for some δ' multiple of δ_1 and δ_2 such that t_1 and t_2 are multiples of δ',
   (a) $\mathcal{M}_1$ is DP-enabled in relation to $Act^I_1 \cap Act^O_2$ and δ'
   (b) $\mathcal{M}_2$ is DP-enabled in relation to $Act^I_2 \cap Act^O_1$ and δ'
   (c) both $\mathcal{M}_1$ and $\mathcal{M}_2$ are δ'-cooperative in relation to $Act_1 \cap Act_2$

then $\mathcal{M}$ is feasible in relation to $J \setminus Act^O_2$ and δ'_1.

Proof. On the one hand, Theo. 4.1 ensures that $\mathcal{M}$ can make independent progress and is initializable. On the other hand, Lem. 4.2 ensures that $\mathcal{M}$ is DP-enabled in relation to $J \setminus Act^O_2$ and δ'_1. Hence, by Prop. 4.3 we can conclude that $\mathcal{M}$ is feasible in relation to $J \setminus Act^O_2$ and δ'_1. □

Approximation of networks by timed machines

Proof of Theo. 5.1

Let $\mathcal{M}_i @ \eta_i$ – $i = 1, 2$ – be such that $\mathcal{M}_1$ and $\mathcal{M}_2$ are compatible and let δ be the greatest common divisor of their clock granularities. The composition

$$\mathcal{M} = {}_{\delta}\Big|\Big|_{i=1,2}^{(l_{\eta_i}, v_{\eta_i}, t_{\eta_i})} \mathcal{M}_i$$

is the machine that best approximates $\Lambda = \iota_1(\Lambda_{\mathcal{M}_1@\eta_1}) \cap \iota_2(\Lambda_{\mathcal{M}_2@\eta_2})$, i.e.,

— $\Lambda_{\mathcal{M}} \precapprox \Lambda$ and,
— for any other machine $\mathcal{M}'$ such that $\Lambda_{\mathcal{M}'} \precapprox \Lambda$, $\mathcal{M}' \precapprox \mathcal{M}$.

Proof. Recall that ι_i is the inclusion $Act_i \subseteq Act_1 \cup Act_2$. Moreover, we have that $\iota_i(\Lambda_{\mathcal{M}_i@\eta_i}) = \{\langle \sigma, \tau\rangle : \langle \sigma|_{\iota_i}, \tau\rangle \in \Lambda_{\mathcal{M}_i@\eta_i}\}$. Let δ be the greatest common divisor of δ_1 and δ_2. By definition, $\mathcal{M} = \|_{i=1,2}^{((l_{\eta_i},0),v_{\eta_i},t_{\eta_i})} \mathcal{M}_{ik_i}$ where $k_i = \delta_i/\delta$. From Def. 3.3, $\mathcal{M} = \langle \delta, \mathcal{A}\rangle$, where $\mathcal{A}$ is the TIOA we obtain by replacing the initial location and clock valuation of $\mathcal{A}_1 \parallel \mathcal{A}_2$ with (l_{η_1}, l_{η_2}) and $v_{\eta_1} \cup v_{\eta_2}$, respectively.

$\mathcal{M} \succeq \Lambda$ — Let λ be a timed trace of $\Lambda_{\mathcal{M}}$. By definition, λ is a refinement of a timed trace λ' generated by an execution β of $\mathcal{M}$ (i.e., its transitions are labelled with the actions in λ'). For $i = 1, 2$, let β_i be the projection of β over $\mathcal{M}_{ik_i}$, i.e., the execution that is obtained by forgetting the locations, the actions and the clocks of the other machine. From Def. 3.3, we can conclude that each is an execution of the corresponding machine starting at the state $((l_{\eta_i}, 0), v_{\eta_i})$.

Because β_i is an execution of $\mathcal{M}_{ik_i}$ starting at the state $((l_{\eta_i}, 0), v_{\eta_i})$, we can extract an execution β_i' of $\mathcal{M}_i$ by forgetting the $k_i - 1$ intermediate edges that make $\mathcal{A}_{ik_i}$ go from every location $(l, 0)$ to $(l, k_i - 1)$, i.e., we retain the edge that makes the transition from $(l, k_i - 1)$ to $(l', 0)$ — this edge also makes a transition from l to l' in $\mathcal{M}_i$. This execution of $\mathcal{M}_i$ starts at the state (l_{η_i}, v_{η_i}). Because the edges that we forget are labelled with $\emptyset$, it is easy to see that λ' is a refinement of the timed trace extracted from the execution β_i' and, therefore, belongs to $\iota_i(\Lambda_{\mathcal{M}_i@\eta_i})$ because $\Lambda_{\mathcal{M}_i@\eta_i}$ is r-closed.

Therefore $\lambda' \in \iota_i(\Lambda_{\mathcal{M}_i@\eta_i})$ for $i = 1, 2$, i.e., $\lambda' \in \Lambda$.

$\mathcal{M} \succapprox \Lambda$ — Let $\lambda = \langle \sigma, \tau\rangle \in \Lambda$. We need to prove that there exists $\lambda' \in \Lambda_{\mathcal{M}}$ that refines λ.

Let $i = 1, 2$. By definition, $\lambda|_{\iota_i}$ is a refinement of a timed trace $\lambda_i = \langle \sigma_i, \tau_{\delta_i}\rangle$ generated by an execution of $\mathcal{M}_i$ starting at the state (l_{η_i}, v_{η_i}). That execution refines to an execution of $\mathcal{M}_{ik_i}$ starting at the state $((l_{\eta_i}, 0), v_{\eta_i})$, which generates the refinement λ_{ik_i} of λ_i over the δ-time sequence (Def. 3.4).

From Defs. 2.7 and 3.4, the corresponding execution of $\mathcal{M}$ starts at state $(((l_{\eta_1}, 0), (l_{\eta_1}, 0)), v_{\eta_1} \cup v_{\eta_2})$ and generates the δ-timed trace λ'' that consists of the pointwise union of all the $\iota_i(\lambda_{ik_i})$ — $\lambda''(n) = \langle n.\delta, \iota_1(\lambda_{1k_1})(n) \cup \iota_2(\lambda_{2k_2})(n)\rangle$.

On the other hand, for every i and n, $\lambda|_{\iota_i}(n.k_i) = \iota_i^{-1}(\lambda''(n.k_i)) = \lambda_i(n)$. Therefore, $\lambda''|_{\iota_i}$ is a refinement of λ_i and, hence, $\lambda'' \in \Lambda \cap \Lambda_{\mathcal{M}}$.

Notice, however, that λ'' is not necessarily a refinement of λ: λ'' is a δ-timed trace, which λ does not need to be.

We now prove that the refinement λ' of λ'' over the meet τ' of τ and τ_δ refines λ — that is, we add to λ'' all the instants present in τ but not in τ_δ labelled with $\emptyset$. We need to prove that λ is also labelled with $\emptyset$ on those instants, that it coincides with

λ'' on the instants that they share, and that λ'' is labelled with $\emptyset$ on those instants of time that are not shared with λ.

Consider an arbitrary instant $\tau'(n)$:

—If $\tau'(n)$ is has been added to τ_δ, this is because it belongs to τ — say, it is $\tau(j)$ for $j = 1$ or $j = 2$ — and is not a multiple of δ. Then, by the definition of refinement, $\lambda'(n) = \emptyset$. On the other hand, $\lambda(j) = \emptyset$, otherwise none of the $\lambda|_{\iota_i}$ could be a refinement of a timed trace generated by $\mathcal{M}_i$ — $\lambda(j) \neq \emptyset$ only when one of the machines makes a move.

—If $\tau'(n)$ is a multiple of δ — say, $\tau'(n) = k.\delta$ — then, by construction, $\lambda'(n) = \lambda''(k) = \iota_1(\lambda_{1k_1})(k) \cup \iota_2(\lambda_{2k_2})(k)$.

–If k is not a multiple of k_i for either $i = 1$ or $i = 2$, then $\tau'(n)$ does not belong to τ. On the other hand, this means that $\lambda''(l) = \emptyset$ because $\tau'(n)$ corresponds to one of the 'ticks' introduced by the refinement of the two machines over δ.

–If $k = l.k_i$ for some for either $i = 1$ or $i = 2$, then it belongs to τ as τ records the instants at which the machines make a move. Hence, λ coincides with λ' at that instant of time.

Finally, because $\lambda'' \in \Lambda_{\mathcal{M}}$ and $\Lambda_{\mathcal{M}}$ is r-closed, $\lambda' \in \Lambda_{\mathcal{M}}$.

$\mathcal{M}$ is the machine that best approximates Λ — We have to prove that for any other machine $\mathcal{M}'$ such that $\Lambda_{\mathcal{M}'} \precapprox \Lambda$, $\mathcal{M}' \precapprox \mathcal{M}$.

We start by proving that $\delta_{\mathcal{M}'}$ is a common divisor of δ_1 and δ_2 and, hence, δ is a multiple of $\delta_{\mathcal{M}'}$.
Let λ a $\delta_{\mathcal{M}'}$-timed trace in $\Lambda_{\mathcal{M}'}$. There must be a $\lambda' \in \Lambda$ that is refined by $\lambda' = \langle \sigma, \tau \rangle$. By definiton of Λ, τ contains all the instants of time multiples of δ_1 and all the instants of time multiples of δ_2 and, hence, $\mathcal{M}'$ needs to generate all those 'ticks' and, hence, $\delta_{\mathcal{M}'}$ needs to divide δ_1 and δ_2.

Let $\lambda \in \Lambda_{\mathcal{M}'}$. Then, by definition, there exists a $\delta_{\mathcal{M}'}$-timed trace $\lambda' \in \Lambda_{\mathcal{M}'}$ that λ refines. Because $\mathcal{M}'$ approximates Λ, there exists a $\lambda^* \in \Lambda$ that λ' refines. Because $\mathcal{M}$ also approximates Λ, there exists a $\lambda''' \in \Lambda_{\mathcal{M}}$ that refines λ^*. By definition, then there exists a $\delta_{\mathcal{M}'}$-timed trace λ'' that is refined by λ'''. Because δ is a multiple of $\delta_{\mathcal{M}'}$, it is not difficult to conclude that λ' refines λ'' and, hence, λ refines λ''.

Let $\lambda \in \Lambda_{\mathcal{M}}$. Because $\mathcal{M}$ approximates Λ, there exists a $\lambda^* \in \Lambda$ that is refined by λ. Because $\mathcal{M}'$ approximates Λ, there exists a $\lambda' \in \Lambda_{\mathcal{M}'}$ that refines λ^*. It is not difficult to conclude that the refinement of λ' over the meet of τ and τ' refines λ.

□

Proof of Prop. 5.1

If $\mathcal{M} \precapprox \alpha$ then $\delta_{\mathcal{M}}$ is a common divisor of all the clock granularities of the nodes of α.

Proof. Let $\Lambda_{\mathcal{M}} \precapprox \Lambda_\alpha$ and $\lambda = \langle \sigma, \tau \rangle \in \Lambda_\alpha$. By definition of $\precapprox$, there must be an execution of $\mathcal{M}$ that generates a timed trace that refines λ. Because τ contains all the instants

of time at which the machines at the nodes 'tick', $\mathcal{M}$ needs to generate all those 'ticks' and, hence, $\delta_{\mathcal{M}}$ needs to divide all the clock granularities of the nodes of α. □

Proof of Theo. 5.2

Let α be a network such that a common divisor of all the clock granularities of its nodes exists and δ_α be their greatest common divisor. The machine

$$\mathcal{M} = \Big|\Big|_{\delta_\alpha}^{(l_{\eta_p}, \iota_p(v_{\eta_p}), t_{\eta_p})}_{p \in N_\alpha} \iota_p(\mathcal{M}_p)$$

where the ι_p translate the alphabet of the automata α_p to Act_α (Def. 5.5) and prefix all clocks of the automata with $p.$, is the best approximation of α.

Proof. This is a generalisation of Theo. 5.1; the result follows from the associativity of composition and intersection. □

Consistency and feasibility of networks

Proof of Prop. 5.6

If a connected network α makes independent progress, then it is consistent. If α is also DP-enabled in relation to each of its interaction-points, then it is feasible.

Proof. We start noticing that in a connected network α, for every node p of α, the machine $\mathcal{M}_p$ can stay at the state (l_{η_p}, v_{η_p}) at least for δ_p time units (by definition of partial execution, the invariant of location l_{η_p} holds for δ_p time units). Moreover, there is a common divisor δ of all the clock granularities δ_p of the nodes of α. Let δ_α be their greatest common divisor. We then have that every $\mathcal{M}_p$ can stay at the state (l_{η_p}, v_{η_p}) at least for δ_α. Because the invariant of the location of the initial state of $\mathcal{M}_\alpha$ is the conjunction of the invariants of each l_{η_p} and the initial clock valuation is $\bigcup_p v_p$, we can easily conclude that $\mathcal{M}_\alpha$ can stay at its initial state at least for δ_α time units and, hence, $\mathcal{M}_\alpha$ is initializable.

From Prop. 4.2, it follows immediately that if α makes independent progress, then it is consistent. From Prop. 4.3 it follows that if α is also DP-enabled in relation to each of its interaction-points, then it is feasible. □

Auxiliary Definition and Result

In order to prove the results for networks built through intra-binding we define two new operation over machines and prove the preservation of some machine properties under this new operation.

Definition 7.1. Given a machine $\mathcal{M}$ and a state (l, v) reachable in $\mathcal{M}$, we define $\mathcal{M} \gg (l, v)$ as the machine that only differs from $\mathcal{M}$ in the initial location and clock valuation which are l and v, respectively. Similarly, given an automata $\mathcal{A}$ we use $\mathcal{A} \gg (l, v)$ to denote the automata that only differs from $\mathcal{A}$ in the initial location and clock valuation which are l and v.

Lemma 7.3. Let $\mathcal{M}$ be a machine and (l, v) a state reachable in $\mathcal{M}$. Let δ' be a multiple of $\delta_{\mathcal{M}}$ such that (l, v) is reachable in $\mathcal{M}$ at a time multiple of δ'.

(1) $\mathcal{M} \gg (l, v)$ is initializable.
(2) if $\mathcal{M}$ makes independent progress, so does $\mathcal{M} \gg (l, v)$.
(3) if $\mathcal{M}$ is DP-enabled in relation to J and δ', so is $\mathcal{M} \gg (l, v)$.
(4) if $\mathcal{M}$ is cooperative in relation to J and δ', so is $\mathcal{M} \gg (l, v)$.

Proof.

(1) It follows trivially by the definition of a reachable state that the machine can stay in (l, v) at least $\delta_{\mathcal{M}}$ time units.

(2) It follows from the fact that all reachable states in $\mathcal{M} \gg (l, v)$ were already reachable in $\mathcal{M}$.

(3) It follows from the fact that states reachable in $\mathcal{M} \gg (l, v)$ at time T are reachable in $\mathcal{M}$ at time $T + k.\delta'$, for some natural number k.

(4) As in the previous case.

□

Definition 7.2. Given a machine $\mathcal{M}$, $J_1 \subseteq Act^I_{\mathcal{M}}$, $J_2 \subseteq Act^O_{\mathcal{M}}$ and a bijection $\xi : J_1 \leftrightarrow J_2$, we define $\xi(\mathcal{M})$ as the machine that only differs from $\mathcal{M}$ in the alphabet and set of edges, which are as follows:

— $Act^I_{\xi(\mathcal{M})} = Act^I_{\mathcal{M}} \setminus J_1$,
— $Act^O_{\xi(\mathcal{M})} = Act^O_{\mathcal{M}} \setminus J_2$,
— $Act^\tau_{\xi(\mathcal{M})} = Act^\tau_{\mathcal{M}} \cup \{\{j_1, j_2\} : j_1 \xi j_2\}$,
— $E_{\xi(\mathcal{M})} = \{(l, \xi^*(S), C, R, l') : (l, S, C, R, l') \in E_{\mathcal{M}}$ and $\xi^*(S)$ is defined$\}$ where

$$\xi^* : 2^{Act_{\mathcal{M}}} \nrightarrow 2^{Act_{\xi(\mathcal{M})}}$$

is the partial function defined only for the sets S such that, for every $j_1 \in J_1$ and $j_2 \in J_2$, $j_1 \xi j_2$ iff $j_1 \in S$ and $j_2 \in S$, in which case $\xi^*(S) = \{\{j_1, j_2\} : j_1 \xi j_2 \wedge j_1 \in S \wedge j_2 \in S\} \cup S \setminus (J_1 \cup J_2)$.

Lemma 7.4. Let $\mathcal{M}$ be a machine and $\xi : J_1 \leftrightarrow J_2$ a bijection with $J_1 \subseteq Act^I_{\mathcal{M}}$ and $J_2 \subseteq Act^O_{\mathcal{M}}$. If $\mathcal{M}$ is initializable and makes independent progress and, for some δ' multiple of $\delta_{\mathcal{M}}$, $\mathcal{M}$ is DP-enabled in relation to $J_1 \subseteq Act^I_{\mathcal{M}}$ and δ' and is cooperative in relation to $J_2 \subseteq Act^O_{\mathcal{M}}$ and δ', then $\xi(\mathcal{M})$ is initializable and makes independent progress. Moreover,

— if $\mathcal{M}$ is DP-enabled in relation to $J \subseteq Act^I_{\mathcal{M}} \setminus J_1$ and δ'' multiple of $\delta_{\mathcal{M}}$, so is $\xi(\mathcal{M})$,
— if $\mathcal{M}$ is cooperative in relation to $J \subseteq Act^O_{\mathcal{M}} \setminus J_2$ and δ'' multiple of $\delta_{\mathcal{M}}$, so is $\xi(\mathcal{M})$.

Proof. The set of initial locations, clock valuations and location invariants of $\xi(\mathcal{M})$ is the same of $\mathcal{M}$ and, hence, if $\mathcal{M}$ is initializable so is $\xi(\mathcal{M})$. For proving that the ability to make progress is also preserved we start noticing that $\xi(\mathcal{M})$ has no more edges than $\mathcal{M}$ and, hence, the states that are reachable in $\xi(\mathcal{M})$ were also reachable in $\mathcal{M}$. Let (l, v) be a state in $\xi(\mathcal{M})$ reachable at time T. Because it is also reachable at time T in $\mathcal{M}$ and this machine makes progress, we know that there exists an edge $(l, A, C, R, l') \in E_{\mathcal{M}}$ such that

(a) $A \subseteq Act^O_{\mathcal{M}} \cup Act^\tau_{\mathcal{M}}$, (b) $v+\delta_{\mathcal{M}} \vDash C$ and (c) for all $0 \leq t \leq \delta_{\mathcal{M}}$, $(v+\delta_{\mathcal{M}})^{\mathbf{R}}+t \vDash Inv(l')$. If $\xi^*(A)$ is defined then $(l, \xi^*(A), C, R, l')$ is an edge of $\xi(\mathcal{M})$ and we have nothing else to prove. If $\xi^*(A)$ is not defined, we consider the biggest set S_2 such that $S_2 \subseteq A \cap (J_2)$ and $\xi(S_2) \not\subseteq A$. We know that this set is not empty. We have two cases:

(1) If $T+\delta_{\mathcal{M}}$ is multiple of δ', then because $\mathcal{M}$ is DP-enabled in relation to J_1 and δ', we know that there exists an edge $(l, A \cup \xi(S_2), C', R', l'') \in E_{\mathcal{M}}$ such that (a) $v+\delta_{\mathcal{M}} \vDash C'$ and (b) for all $0 \leq t \leq \delta_{\mathcal{M}}$, $(v+\delta_{\mathcal{M}})^{\mathbf{R}'} + t \vDash Inv(l'')$. Let $A' = A \cup \xi(S_2)$. By construction of A' we know that $\xi^*(A')$ is defined and, hence, (l, A', C', R', l'') is an edge of $\xi(\mathcal{M})$. We also know that $\xi(S_2) \subseteq Act^\tau_{\xi(\mathcal{M})}$ and, hence, $A' \subseteq Act^O_{\xi(\mathcal{M})} \cup Act^\tau_{\xi(\mathcal{M})}$.

(2) If $T + \delta_{\mathcal{M}}$ is not a multiple of δ', then because $\mathcal{M}$ is cooperative in relation to J_2 and δ', we know that there exists an edge $(l, A \setminus S_2, C', R', l'') \in E_{\mathcal{M}}$ such that (a) $v+\delta_{\mathcal{M}} \vDash C'$ and (b) for all $0 \leq t \leq \delta_{\mathcal{M}}$, $(v+\delta_{\mathcal{M}})^{\mathbf{R}'}+t \vDash Inv(l'')$. Let $A' = A \setminus S_2$. By construction of A' we know that $\xi^*(A')$ is defined and, hence, (l, A', C', R', l'') is an edge of $\xi(\mathcal{M})$. We also know that $A' \subseteq Act^O_{\xi(\mathcal{M})} \cup Act^\tau_{\xi(\mathcal{M})}$.

We now prove the second part. Suppose that $\mathcal{M}$ is DP-enabled in relation to $J \subseteq Act^I_{\mathcal{M}} \setminus J_1$ and δ'' multiple of $\delta_{\mathcal{M}}$. Let (l, v) be a state in $\xi(\mathcal{M})$ reachable at time T such that $(T+\delta_{\mathcal{M}})$ is a multiple of δ'' and $B \subseteq J$. Also, let (l, A, C, R, l') be an edge in $E_{\xi(\mathcal{M})}$ such that $v + \delta_{\mathcal{M}} \vDash C$ and, for all $0 \leq t \leq \delta_{\mathcal{M}}$, $(v + \delta_{\mathcal{M}})^{\mathbf{R}} + t \vDash Inv_{\mathcal{M}}(l')$. By construction of $\xi(\mathcal{M})$ we have that $(l, \xi^{*-1}(A), C, R, l')$ be an edge in $E_{\xi(\mathcal{M})}$. Because (l, v) is also reachable at time T in $\mathcal{M}$ and this machine is DP-enabled in relation to $J \subseteq Act^I_{\mathcal{M}} \setminus J_1$ and δ'' we have that there exists an edge $(l, B \cup (\xi^{*-1}(A) \setminus J), C', R', l'')$ in $E_{\mathcal{M}}$ such that $v + \delta_{\mathcal{M}} \vDash C'$ and, for all $0 \leq t \leq \delta_{\mathcal{M}}$, $(v+\delta_{\mathcal{M}})^{\mathbf{R}'} + t \vDash Inv_{\mathcal{M}}(l'')$. Because $J \subseteq Act^I_{\mathcal{M}} \setminus J_1$ and $B \subseteq J$, $\xi^*(B \cup (\xi^{*-1}(A) \setminus J) = B \cup A \setminus J$. Hence, we the have that $(l, \xi^*(B \cup (A \setminus J)), C', R', l'')$ is in $E_{\xi(\mathcal{M})}$. The proof of preservation of cooperativeness in relation to $J \subseteq Act^O_{\mathcal{M}} \setminus J_2$ follows the same lines. □

Proof of Prop. 5.3

Let α be a connected network and $\xi_{\{p,q\}} : M_p \leftrightarrow M_q$ be an attachment where $\langle p, M_p \rangle$ and $\langle q, M_q \rangle$ are interaction points such that δ_p and δ_q are commensurate.

(1) If α makes independent progress and is DP-enabled in relation to the interaction points $\langle p, M_p \rangle$ and $\langle q, M_q \rangle$ and, for some δ common multiple of δ_p and δ_q, α is both δ-cooperative in relation to $\langle p, M_p \rangle$ and $\langle q, M_q \rangle$, then $\alpha^+_{\xi_{\{p,q\}}}$ also makes independent progress and, therefore, is consistent.

Moreover, for every interaction point $\langle v, M_v \rangle$ of α different from those involved in the attachment:

(2) If α is DP-enabled in relation to $\langle v, M_v \rangle$ so is $\alpha^+_{\xi_{\{p,q\}}}$

(3) If α is cooperative in relation to $\langle v, M_v \rangle$ and δ multiple of δ_v so is $\alpha^+_{\xi_{\{p,q\}}}$.

Therefore, under the conditions of (1) and (2) above, we can conclude that, for every interaction point $\langle v, M_v \rangle$ of α different from those involved in the attachment, $\alpha^+_{\xi_{\{p,q\}}}$ is feasible in relation to $\langle v, M_v \rangle$ if so is α .

Proof. We start noticing that $\xi_{\{p,q\}} : M_p \leftrightarrow M_q$, being an attachment, also defines a bijection between $\langle p, M_p\rangle^I \cup \langle q, M_q\rangle^I$ and $\langle p, M_p\rangle^O \cup \langle q, M_q\rangle^O$. We also use $\xi_{\{p,q\}}$ to denote this bijection. Let N be the set of nodes of α, which is also the set of nodes of $\alpha^+_{\xi_{\{p,q\}}}$.

(1) For every $v \in N$, let η'_v be the partial execution of $\mathcal{M}_v$ at which the binding is made whose first state is the last state of η_v. We first prove that

$$\mathcal{M}_{\alpha^+_{\xi_{\{p,q\}}}} = \xi_{\{p,q\}}(\mathcal{M}_\alpha \gg (l', v'))$$

where $l' = \otimes_{v\in N}(l_{\eta_v;\eta'_v}, 0)$ and $v' = \bigcup_{v\in N} v_{\eta_v;\eta'_v}$.

On the one hand we have that

$$\mathcal{M}_\alpha = {}_{\delta_\alpha}\Big|\Big|^{(l_{\eta_v},\iota_v(v_{\eta_v}),t_{\eta_v})}_{v\in N} \iota_v(\mathcal{M}_v) = \langle \delta_\alpha, \Big|\Big|_{v\in N} \iota_v(\mathcal{A}_{v_{k_v}}) \gg (l, v)\rangle$$

where δ_α is the greatest common divisor of $\{\delta_v : v \in N\}$, $k_v = \delta_v/\delta_\alpha$, ι_v translates the alphabet of the automata $\mathcal{A}_{v_{k_v}}$ to Act_α and translates every clock c of the automata $\mathcal{A}_{v_{k_v}}$ to $v.c$ and $l = \otimes_{v\in N}(l_{\eta_v}, 0)$ and $v = \bigcup_{v\in N} v_{\eta_v}$. Recall that for the messages in ports not connected in α, such as $\langle p, M_p\rangle$ and $\langle q, M_q\rangle$, $\iota_v(a) = v.a$.

Because there is a δ_α-time prefix π of Π_α such that the projection of π to the behaviour of each $\mathcal{M}_p$ refines the prefix generated by η'_p, we can conclude that (l', v') is a state reachable in $\mathcal{M}_\alpha$ in a time that is both multiple of δ_p and δ_q.

$$\mathcal{M}_\alpha \gg (l', v') = \langle \delta_\alpha, \Big|\Big|_{v\in N} \iota_v(\mathcal{A}_{v_{k_v}}) \gg (l', v')\rangle$$

Then we have that

$$\xi_{\{p,q\}}(\mathcal{M}_\alpha \gg (l', v')) = \langle \delta_\alpha, \xi_{\{p,q\}}(\Big|\Big|_{v\in N} \iota_v(\mathcal{A}_{v_{k_v}})) \gg (l', v')\rangle$$

On the other hand, we have that

$$\mathcal{M}_{\alpha^+_{\xi_{\{p,q\}}}} = {}_{\delta_\alpha}\Big|\Big|^{(l_{\eta_v;\eta'_v},\iota'_v(v_{\eta_v;\eta'_v}),t_{\eta_v;\eta'_v})}_{v\in N} \iota'_v(\mathcal{M}_v) = \langle \delta_\alpha, \Big|\Big|_{v\in N} \iota'_v(\mathcal{A}_{v_{k_v}}) \gg (l', v')\rangle$$

where ι'_v translates the alphabet of the automata $\mathcal{A}_{v_{k_v}}$ to the alphabet of $\alpha^+_{\xi_{\{p,q\}}}$ and translates every clock c to $v.c$. Recall that messages in all ports but $\langle p, M_p\rangle$ and $\langle q, M_q\rangle$ are translated by ι'_v in the same way they are translated by ι_v whereas messages $a \in M_p$ are translated by ι'_p to $\{p.a, q.\xi_{\{p,q\}}(a)\}$ and messages in $a \in M_q$ are translated by ι'_q to $\{q.a, p.\xi_{\{p,q\}}(a)\}$.

Because the application of $\xi_{\{p,q\}}$ to $||_{v\in N}\, \iota_v(\mathcal{A}_{v_{k_v}})$ replaces actions of the form $p.a$ and $q.a$ by internal actions $\{p.a, q.\xi_{\{p,q\}}(a)\}$ and $\{q.a, p.\xi_{\{p,q\}}(a)\}$, respectively, it is easy to conclude that the alphabet of $\xi_{\{p,q\}}(\mathcal{M}_\alpha)$ is equal to that of $\mathcal{M}_{\alpha^+_{\xi_{\{p,q\}}}}$.

Now we just need to prove that the two automata also have the same edges.

(a) $(\otimes_{v\in N} q_v, S, C, R, \otimes_{v\in N} q'_v)$ is an edge of $||_{v\in N}\, \iota_v(\mathcal{A}_{v_{k_v}})$ iff $C = \wedge_{v\in N} C_v$, $R =$

$\cup_{v\in N} R_v$ and $(q_v, S \cap \iota_v(Act_v), C_v, R_v, q'_v)$ is an edge of $\iota_v(\mathcal{A}_{v_{k_v}})$, which is equivalent to $(q_v, \iota_v^{-1}(S \cap \iota_v(Act_v)), \iota_v^{-1}(C_v), \iota_v^{-1}(R_v), q'_v)$ is an edge of $\mathcal{A}_{v_{k_v}}$.

(b) Then, by construction of $\xi_{\{p,q\}}(||_{v\in N}\, \iota_v(\mathcal{A}_{v_{k_v}}))$, we have that $(\otimes_{v\in N} q_v, S, C, R, \otimes_{v\in N} q'_v)$ is an edge of $\xi_{\{p,q\}}(||_{v\in N}\, \iota_v(\mathcal{A}_{v_{k_v}}))$ iff

$$C = \wedge_{v\in N} C_v \text{ and } R = \cup_{v\in N} R_v \text{ and}$$

$$(q_v, \iota_v^{-1}(\xi_{\{p,q\}}^{*^{-1}}(S) \cap \iota_v(Act_v)), \iota_v^{-1}(C_v), \iota_v^{-1}(R_v), q'_v) \text{ is an edge of } \mathcal{A}_{v_{k_v}} (*)$$

(c) We need to conclude $(*)$ is equivalent to $(\otimes_{v\in N} q_v, S, C, R, \otimes_{v\in N} q'_v)$ is an edge of $||_{v\in N}\, \iota'_v(\mathcal{A}_{v_{k_v}})$. We know that this condition is equivalent to $C = \wedge_{v\in N} C_v$, $R = \cup_{v\in N} R_v$ and $(q_v, \iota'^{-1}_v(S \cap \iota'_v(Act_v)), \iota'^{-1}_v(C_v), \iota'^{-1}_v(R_v), q'_v)$ is an edge of $\mathcal{A}_{v_{k_v}}$. Because ι_v and ι'_v translate clocks in the same way, we only need to prove that

$$\iota_v^{-1}(\xi_{\{p,q\}}^{*^{-1}}(S) \cap \iota_v(Act_v)) = \iota'^{-1}_v(S \cap \iota'_v(Act_v))$$

—For actions $a \notin M_p \cup M_q$:

– $a \in \iota_v^{-1}(\xi_{\{p,q\}}^{*^{-1}}(S) \cap \iota_v(Act_v))$ iff
$\iota_v(a) \in \xi_{\{p,q\}}^{*^{-1}}(S)$ and $\iota_v(a) \in \iota_v(Act_v)$ iff
$v.a \in \xi_{\{p,q\}}^{*^{-1}}(S)$ and $a \in Act_v$ iff
$v.a \in S$ and $a \in Act_v$.

– $a \in \iota'^{-1}_v(S \cap \iota'_v(Act_v))$ iff
$\iota'_v(a) \in S$ and $\iota'_v(a) \in \iota'_v(Act_v)$ iff
$v.a \in S$ and $a \in Act_v$.

—For actions $a \in M_p$:

– $a \in \iota_v^{-1}(\xi_{\{p,q\}}^{*^{-1}}(S) \cap \iota_v(Act_v))$ iff
$\iota_v(a) \in \xi_{\{p,q\}}^{*^{-1}}(S)$ and $\iota_v(a) \in \iota_v(Act_v)$ iff
$\{p.a, q.\xi_{\{p,q\}}(a)\} \in \xi_{\{p,q\}}^{*^{-1}}(S)$ and $a \in Act_v$ iff
$\{p.a, q.\xi_{\{p,q\}}(a)\} \in S$ and $a \in Act_v$.

– $a \in \iota'^{-1}_v(S \cap \iota'_v(Act_v))$ iff
$\iota'_v(a) \in S$ and $\iota'_v(a) \in \iota'_v(Act_v)$ iff
$\{p.a, q.\xi_{\{p,q\}}(a)\} \in S$ and $a \in Act_v$.

—For actions $a \in M_q$ the proof is similar.

(2) Second we notice that we are in conditions of applying Lemma 7.3: (l', v') is reachable in $\mathcal{M}_\alpha$ at a time multiple of δ_p and δ_q (and, hence, also of δ_α). Hence, we can conclude that $\mathcal{M}_\alpha \gg (l', v')$ is initializable and makes independent progress, is DP-enabled in relation to $\langle p, M_p\rangle^I$ and δ_p and in relation to $\langle q, M_q\rangle^I$ and δ_q and, for some δ common multiple of δ_p and δ_q, $\mathcal{M}_\alpha \gg (l', v')$ is both δ-cooperative in relation to $\langle p, M_p\rangle^O$ and $\langle q, M_q\rangle^O$.

(3) Third, we show that we are in the conditions of applying Lemma 7.4. We know that $\mathcal{M}_\alpha \gg (l', v')$ is DP-enabled in relation to both $\langle p, M_p\rangle^I$ and δ_p and to $\langle q, M_q\rangle^I$ and δ_q. This implies that $\mathcal{M}_\alpha \gg (l', v')$ is DP-enabled in relation to $\langle p, M_p\rangle^I \cup \langle q, M_q\rangle^I$ and any multiple of $\delta_p \cdot \delta_q$. Similarly, the fact that, for some δ common multiple of δ_p and δ_q, $\mathcal{M}_\alpha$ is δ-cooperative in relation to both $\langle p, M_p\rangle^O$ and $\langle q, M_q\rangle^O$, implies that $\mathcal{M}_\alpha \gg (l', v')$ is δ-cooperative in relation to $\langle p, M_p\rangle^O \cup \langle q, M_q\rangle^O$. We can then conclude that $\mathcal{M}_{\alpha^+_{\xi_{\{p,q\}}}}$ is initializable and makes independent progress.

(4) We now prove that, if $\mathcal{M}_\alpha$ is DP-enabled in relation to $\langle v, M\rangle^I$ and δ_v and $\langle v, M\rangle$ is an interaction point of $\mathcal{M}_\alpha$ different from those being connected, so is $\mathcal{M}_{\alpha^+_{\xi_{\{p,q\}}}}$. Because the intersection of $\langle v, M\rangle^I$ with $\langle p, M_p\rangle^I \cup \langle q, M_q\rangle^I$ is empty, we can use the second part of Lemma 7.4 to conclude that $\mathcal{M}_{\alpha^+_{\xi_{\{p,q\}}}}$ is DP-enabled in relation to $\langle v, M\rangle^I$ and δ_v. In a similar way, we can prove that if $\mathcal{M}_\alpha$ is cooperative in relation to $\langle v, M\rangle^O$ and δ multiple of δ_v and $\langle v, M\rangle$ is an interaction point different from those being connected, so is $\mathcal{M}_{\alpha^+_{\xi_{\{p,q\}}}}$.

□

Proof of Theo. 5.4

Let α_1 and α_2 be two disjoint connected networks and $\xi_{\{p,q\}} : M_p \leftrightarrow M_q$ be an attachment where $\langle p, M_p\rangle$ is an interaction point of α_1 and $\langle q, M_q\rangle$ an interaction point of α_2 such that δ_p and δ_q are commensurate.

(1) If α_1 and α_2 make independent progress, α_1 is DP-enabled in relation to $\langle p, M_p\rangle$, α_2 is DP-enabled in relation to $\langle q, M_q\rangle$, and for some δ common multiple of δ_p and δ_q, α_1 is cooperative in relation to $\langle p, M_p\rangle$ and δ, and α_2 is cooperative in relation to $\langle q, M_q\rangle$ and δ, then $(\alpha_1\ {}_{\xi_{\{p,q\}}}\ \alpha_2)$ also makes independent progress and, therefore, is consistent.

Moreover, for every interaction point $\langle v, M_v\rangle$ of α_i different from that involved in the attachment:

(2) If α_i is DP-enabled in relation to $\langle v, M_v\rangle$ so is $(\alpha_1\ {}_{\xi_{\{p,q\}}}\ \alpha_2)$.
(3) If α_i is cooperative in relation to $\langle v, M_v\rangle$ and δ multiple of δ_v so is $(\alpha_1\ {}_{\xi_{\{p,q\}}}\ \alpha_2)$.

Therefore, under the conditions of (1) and (2) above, we can conclude that, for every interaction point $\langle v, M_v\rangle$ of α_i different from that involved in the attachment,$(\alpha_1\ {}_{\xi_{\{p,q\}}}\ \alpha_2)$ is feasible in relation to $\langle v, M_v\rangle$ if so is α_i.

Proof. For every $v \in N_{\alpha_i}$, let η'_v be the partial execution of $\mathcal{M}_v$ at which the binding of α_1 and α_2 is made (recall that, by definition of inter-binding, the first state of η'_v is necessarily the last state of η_v).

On the one hand, we have that

$$\mathcal{M}_{(\alpha_1\ {}_{\xi_{\{p,q\}}}\ \alpha_2)} = {}_{\delta''}\Big|\Big|^{(l_{\eta_v;\eta'_v}, \iota'_v(v_{\eta_v;\eta'_v}), t_{\eta_v;\eta'_v})}_{v \in N_{\alpha_1} \cup N_{\alpha_2}} \iota'_v(\mathcal{M}_v)$$

where δ'' is the greatest common divisor of $\{\delta_v : v \in N_{\alpha_1} \cup N_{\alpha_2}\}$, ι'_v translates the alphabet

of $\mathcal{M}_v$ to the alphabet of $(\alpha_1 \ {}_{\xi_{\{p,q\}}} \ \alpha_2)$ and translates every clock c to $v.c$. Notice that δ'' exists because each network α_i is connected (and therefore has a greatest common divisor δ_{α_i} of its clock granularities) and the fact that δ_p and δ_q are commensurate makes δ_{α_1} and δ_{α_2} commensurate; in fact δ'' is the greatest common divisor of δ_{α_1} and δ_{α_2}.

On the other hand, for $i = 1, 2$ we have that

$$\mathcal{M}_{\alpha_i} = {}_{\delta_{\alpha_i}}\Big|\Big|^{(l_{\eta_v}, \iota_v(v_{\eta_v}), t_{\eta_v})}_{v \in N_{\alpha_i}} \iota_v(\mathcal{M}_v)$$

where δ_{α_i} is the greatest common divisor of $\{\delta_v : v \in N_{\alpha_i}\}$ and ι_v translates the alphabet of $\mathcal{M}_v$ to Act_{α_i} and translates every clock c to $v.c$.

By definition of inter-binding we have that, for $i = 1, 2$, there is a δ_{α_i}-time prefix π_{α_i} of Π_{α_i} such that, for every $v \in N_{\alpha_i}$, the projection of π_{α_i} to the behaviour of each $\mathcal{M}_v$ refines the prefix generated by η'_v. For $i = 1, 2$, let $l_i = \otimes_{v \in N_{\alpha_i}}(l_{\eta_v;\eta'_v}, 0)$ and $v_i = \bigcup_{v \in N_{\alpha_i}} v_{\eta_v;\eta'_v}$ and t_i be the timestamp of the last action in π_{α_i}. The conditions met by η'_v for every $v \in N_{\alpha_i}$ allow us to conclude that (l_i, v_i) is a state reachable in $\mathcal{M}_{\alpha_i}$ in time t_i. Moreover, we know that t_i is a multiple of δ_v for every $v \in N_{\alpha_i}$.

Because α_1 and α_2 are disjoint, the machines $\mathcal{M}_{\alpha_1}$ and $\mathcal{M}_{\alpha_2}$ have disjoint alphabets and, hence,

$$\mathcal{M} = {}_{\delta'}\Big|\Big|^{(l_i, v_i, t_i)}_{i=1,2} \mathcal{M}_{\alpha_i}$$

where δ' is the greatest common divisor of δ_{α_1} and δ_{α_2}, corresponds to the parallel composition with no communication. In particular, the set of input, output and private actions of $\mathcal{M}$ is obtained through the union of the corresponding sets of actions in $\mathcal{M}_{\alpha_1}$ and $\mathcal{M}_{\alpha_2}$.

We start by noting that, as observed above, δ' and δ'' are the same. Then, we have that $\xi_{\{p,q\}} : M_p \leftrightarrow M_q$, being an attachment, also defines a bijection between $\langle p, M_p\rangle^I \cup \langle q, M_q\rangle^I$ and $\langle p, M_p\rangle^O \cup \langle q, M_q\rangle^O$. Because these are sets of input actions and output actions of $\mathcal{M}$, respectively, we can build $\xi_{\{p,q\}}(\mathcal{M})$. We have that

$$\mathcal{M}_{(\alpha_1 \ {}_{\xi_{\{p,q\}}} \ \alpha_2)} \equiv \xi_{\{p,q\}}(\mathcal{M})$$

where $\equiv$ represents α-equivalence (under locations renaming). The proof of this result follows the same lines of the similar result presented already in the proof of Proposition 5.3. Location renaming is needed in this case because the locations of $\mathcal{M}$ are of the form $((\otimes_{v \in N_{\alpha_1}}(l_v, i), j), (\otimes_{v \in N_{\alpha_2}}(l_v, k), m))$ whereas the locations of $\mathcal{M}_{(\alpha_1 \ {}_{\xi_{\{p,q\}}} \ \alpha_2)}$ are of the form $\otimes_{v \in N_{\alpha_1} \cup N_{\alpha_2}}(l_v, n)$.

By Theorem 4.1 we have that if $\mathcal{M}_{\alpha_1}$ and $\mathcal{M}_{\alpha_2}$ are initializable and make independent progress, so does $\mathcal{M}$. As noticed before, t_i is a multiple of δ_v for every $v \in N_{\alpha_i}$ and, hence, by applying Lemma 4.2 we can conclude that

—If $\mathcal{M}_{\alpha_i}$ is DP-enabled in relation to $\langle v, M_v\rangle^I$ and δ_v, so is $\mathcal{M}$ (because the two machines have disjoint alphabets, the intersection of $\langle v, M_v\rangle^I$ with the set of output actions of the other machine is empty).

—If $\mathcal{M}_{\alpha_i}$ is cooperative in relation to $\langle v, M_v\rangle^O$ and δ a multiple of δ_v, so is $\mathcal{M}$ (because the two machines have disjoint alphabets, $\langle v, M_v\rangle^O$ does not include any input action of the other machine).

In particular, we have that

—$\mathcal{M}$ is DP-enabled in relation to both $\langle p, M_p\rangle^I$ and δ_p and $\langle q, M_q\rangle^I$ and δ_q;
—for some δ common multiple of δ_p and δ_q, $\mathcal{M}$ is δ-cooperative in relation to both $\langle p, M_p\rangle^O$ and $\langle q, M_q\rangle^O$.

This implies that we are in conditions of applying Lemma 7.4. We use the first part of this lemma to conclude that $\xi_{\{p,q\}}(\mathcal{M})$ is initializable and makes independent progress.

For every interaction point $\langle v, M_v\rangle$ of α_i different from that involved in the attachment we have that $\langle v, M_v\rangle$ is still an interaction point of $\xi_{\{p,q\}}(\mathcal{M})$. Because $\mathcal{M}$ is DP-enabled in relation to $\langle v, M_v\rangle^I$ and δ_v and the intersection of $\langle v, M_v\rangle^I$ with $\langle p, M_p\rangle^I \cup \langle q, M_q\rangle^I$ is empty, by the second part of Lemma 7.4, we can conclude that $\xi_{\{p,q\}}(\mathcal{M})$ is DP-enabled in relation to $\langle v, M_v\rangle^I$ and δ_v. Similarly, we can conclude that $\xi_{\{p,q\}}(\mathcal{M})$ is cooperative in relation to $\langle v, M_v\rangle^O$ and δ a multiple of δ_v. □