TBA

MAHIEDDINE DELLABANI, University Grenoble Alpes, VERIMAG
JACQUES COMBAZ, University Grenoble Alpes, VERIMAG
MOHAMED FOUGHAL, LAAS
SADDEK BENSALEM, University Grenoble Alpes, VERIMAG

Design, implementation and verification of distributed real-time systems is acknowledged to be a very hard task. Such systems are prone to different kind of delays, such as execution times of actions or communication delays implied by the distributed platform. The latter increases considerably the complexity of coordinating the parallel activities of running components. In order to ensure global consistency and timing constraints satisfaction, scheduling of such systems must cope with those delays, and propose and execution strategy to avoid execution failure. In this paper, we investigate a formal model for such systems as compositions of timed automata subject to multi-party interactions, and propose a method aiming to overcome the communication delays problem through planning ahead interactions. Moreover, we propose a method to enforce correctness for such systems and provide a compositional verification method for checking dealock-freedom property of the proposed approach.

CCS Concepts: • **Theory of computation → Formal languages and automata theory**; **Logic and verification**; • **Computer systems organization → Embedded and cyber-physical systems**;

Additional Key Words and Phrases: Distributed Real-Time Systems·Timed Automata·Compositional Verification

ACM Reference format:
Mahieddine Dellabani, Jacques Combaz, Mohamed Foughal, and Saddek Bensalem. 2017. TBA. 1, 1, Article 1 (January 2017), 7 pages.
DOI: 10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Nowadays, real-time systems are ubiquitous in several application domains, and such an emergence led to an increasing need of performance: resources, availability, concurrency...etc. This expansion initiates a shift from the use of single processor based hardware platforms, to large sets of interconnected and distributed computing nodes. Moreover, it prompts the birth of a new family of systems known as *Networked Embedded Systems*, and that are intrinsically distributed. Such evolution stems from an increase in complexity of real-time software embedded on such platforms (e.g. electronic control in avionics and automotive domains [14]), and the need to integrate formerly isolated systems [21] so that they can cooperate as well as share resources, improving functionality and reducing costs.

To deal with such complexity, the community of safety critical systems often restricts its scope to predictable systems, which are represented with domain specific models (e.g. periodic tasks, synchronous systems, time-deterministic systems) for which the range of possible executions is small enough to be easily analyzed, allowing the pre-computation of optimal control strategies. *Networked Embedded Systems* usually describes a set of real-time systems, distributed across platform(s) and interacting through a network. Because of their adaptive behavior, the standard practice when implementing such systems is not to rely on models for pre-computation but rather to design systems dynamically adapting at runtime to the actual context of execution. Such approaches do not offer any formal guarantee of timeliness. The lack of a priori knowledge on system behavior leaves also little room for static optimization.

Model-based development is one promising approach in building distributed real-time system. First, an application model is designed, expressing abstraction of the timed system behavior. This abstraction is platform independent, meaning that it does not consider platform introduced delays or CPU speed, which allow to: *(i)* model the system at early stages without any knowledge of the target platform, and *(ii)* verify the obtained model against some safety properties (functional requirements). Thereafter, the application source code, which represents the actual implementation of the system on a given platform, is automatically generated from the model. The big challenge becomes then how to verify the timing behavior of the implementation, since a lot of assumption drops such as atomic execution of action or timeless communication delays.

In this paper, we propose an extension of the work presented in (cite fm) consisting in a model-based approach aiming to mitigate the communication delays of distributed platforms. In this approach, systems consist of components represented as timed automata that may synchronize on particular actions to coordinate their activities. We contribute to this research field by proposing a method for scheduling interactions with bounded horizons aiming to reduce the impact of communication delays on systems execution. We extend our previous work by defining lower bound horizons for planning interactions, which correspond to the platform communication delays. In particular, *(i)* we extend the semantics of (cite fm) for *planning* interactions with lower bound horizons, *(ii)* we discuss its correctness, and *(iii)* provide a formal method for deadlock detection on such models.

The rest of the paper is organized as follows. In Section 2, we provide a definition of timed automata with respect to multiparty interactions. In Section 3, we present our extended planning semantics and discuss its relation with the usual timed automata semantics. Then, we propose a semantical execution strategy to enforce correctness, and finally, provide a compositional verification method for checking the deadlock freedom property of the presented model (Section 4).

DOI: 10.1145/nnnnnnn.nnnnnnn

2 TIMED SYSTEMS AND PROPERTIES

In the framework of the present paper, components are timed automata and systems are compositions of timed automata with respect to multiparty interactions. The timed automata we use are essentially the ones from [2], however, slightly adapted to embrace a uniform notation throughout the paper.

Definition 2.1 (Component). A *component* is a tuple $(\mathcal{L}, \ell_0, \mathcal{A}, \mathcal{T}, \mathcal{X}, tpc)$ where $\mathcal{L}$ is a finite set of *locations*, $\ell_0 \in \mathcal{L}$ is an *initial* location, $\mathcal{A}$ a finite set of *actions*, $\mathcal{X}$ is a finite set of *clocks*, $\mathcal{T} \subseteq \mathcal{L} \times (\mathcal{A} \times C \times 2^{\mathcal{X}}) \times \mathcal{L}$ is a set of *transitions* labeled with an action, a guard, and a set of clocks to be reset, and $tpc : \mathcal{L} \to C$ assigns a *time progress condition*, tpc_ℓ, to each location, where C is the set of *clock constraints* defined by the following grammar:

$$C := true \mid x \sim ct \mid x - y \sim ct \mid C \wedge C \mid false,$$

with $x, y \in \mathcal{X}$, $\sim \in \{<, \leq, =, \geq, >\}$ and $ct \in \mathbb{R}_{\geq 0}$. Time progress conditions are restricted to conjunctions of constraints of the form $x \leq ct$.

Throughout the paper, we consider that components are deterministic timed automata, that is, at a given location ℓ and for a given action a, there is at most one outgoing transition from ℓ labeled by a. Given a timed automaton $(\mathcal{L}, \ell_0, \mathcal{A}, \mathcal{T}, \mathcal{X}, tpc)$, we write $\ell \xrightarrow{a,g,r} \ell'$ if there exists a transition $\tau = \big(\ell, (a, g, r), \ell'\big) \in \mathcal{T}$. We also write:

$$guard(a, \ell) = \begin{cases} g, & \text{if } \exists \tau = \big(\ell, (a, g, r), \ell'\big) \in \mathcal{T} \\ false, & \text{otherwise} \end{cases}$$

Let $\mathcal{V}$ be the set of all clock valuation functions $v : \mathcal{X} \to \mathbb{R}_{\geq 0}$. For a clock constraint C, $C(v)$ is a boolean value corresponding to the evaluation of C on v. For a valuation $v \in \mathcal{V}$, $v + \delta$ is the valuation satisfying $(v + \delta)(x) = v(x) + \delta$, while for a subset of clocks r, $v[r]$ is the valuation obtained from v by resetting clocks of r, i.e. $v[r](x) = 0$ for $x \in r$, $v[r](x) = v(x)$ otherwise. We also denote by $C + \delta$ the clock constraint C shifted by δ, i.e. such that $C(v + \delta)$ iff $C(v)$.

Definition 2.2 (Semantics). A component $B = (\mathcal{L}, \ell_0, \mathcal{A}, \mathcal{T}, \mathcal{X}, tpc)$ defines the labeled transition system (LTS) $(Q, \mathcal{A} \cup \mathbb{R}_{>0}, \to)$ where $Q \subseteq \mathcal{L} \times \mathcal{V}(\mathcal{X})$ denotes the *states* of B, and $\to \subseteq Q \times (A \cup \mathbb{R}_{>0}) \times Q$ denotes the set of transitions between states according to the rules:

- $(\ell, v) \xrightarrow{a} (\ell', v[r])$ if $\ell \xrightarrow{a,g,r} \ell'$, and $g(v)$ is true (action step).
- $(\ell, v) \xrightarrow{\delta} (\ell, v + \delta)$ if $tpc_\ell(v + \delta)$ for $\delta \in \mathbb{R}_{>0}$ (time step).

We define the predicate $urg(tpc_\ell)$ characterizing the urgency of a time progress condition $tpc_\ell = \bigwedge_{i=1}^{m} x_i \leq ct_i$ at a state (ℓ, v) as follows:

$$urg(tpc_\ell) = \bigvee_{i=1}^{m} (x_i = ct_i),$$

An *execution sequence* of B from a state (ℓ, v) is a path in the LTS starting at (ℓ, v) and that alternates action steps and time steps, that is:

$$(\ell_1, v_1) \xrightarrow{\sigma_1} \ldots \xrightarrow{\sigma_i} (\ell_n, v_n), n \in \mathbb{Z}_{>0}, \sigma \in \mathcal{A} \cup \mathbb{R}_{>0}$$

In this paper, we always assume components with *well formed guards* meaning that transitions $\ell \xrightarrow{a,g,r} \ell'$ satisfy $g(v) \Rightarrow tpc_\ell(v) \wedge tpc_{\ell'}(v[r])$ for any $v \in \mathcal{V}$. We say that a state (ℓ, v) is *reachable* if there is an execution sequence from the initial configuration (ℓ_0, v_0) leading to (ℓ, v), where v_0 assigns 0 to all clocks. Notice that the set of reachable states is in general infinite, but it can be partitioned into a finite number of symbolic states [8, 18, 31]. A symbolic state is defined by a pair (ℓ, ζ) where, ℓ is a location of B, and ζ is a zone, i.e. a set of clock valuations defined by a clock constraint (as defined in Definition 2.1). Efficient algorithms for computing symbolic states and operations on zones are fully described in [8]. Given symbolic states $\{(\ell_j, \zeta_j)\}_{j \in J}$ of B, the predicate $Reach(B)$ characterizing the reachable states can be formulated as:

$$Reach(B) = \bigvee_{j \in J} \mathrm{at}(\ell_j) \wedge \zeta_j,$$

where $\mathrm{at}(\ell_j)$ is true on states whose location is ℓ_j, and clock constraint ζ_j is straightforwardly applied to clock valuation functions of states.

We also define the predicate $Enabled(a)$ characterizing states (ℓ, v) at which an action a is enabled, i.e. such that $(\ell, v) \xrightarrow{a} (\ell', v')$. It can be written:

$$Enabled(a) = \bigvee_{(\ell, a, g, r, \ell') \in T} \mathrm{at}(\ell) \wedge guard(a, \ell)$$

Definition 2.3 (Deadlock). We say that a state (ℓ, v) of a component B deadlocks, if neither action steps nor time steps can be done from this state. The following equation characterizes those states:

$$\forall a \in \mathcal{A}.\ \neg Enabled(a) \ \wedge\ urg(tpc_\ell)$$

In our framework, components communicate by means of *multiparty interactions.* A multiparty interaction is a rendez-vous synchronization between actions of a fixed subset of components. It takes place only if all the participants agree to execute the corresponding actions. Given n components B_i, $i = 1, \ldots, n$, with disjoint sets of actions $\mathcal{A}_i$, an interaction is a subset of actions $\alpha \subseteq \cup_{1 \leq i \leq n} \mathcal{A}_i$ containing at most one action per component, i.e. $\alpha \cap \mathcal{A}_i$ is either empty or a singleton $\{a_i\}$. That is, an interaction α can be put in the form $\{a_i\}_{i \in I}$ with $I \subseteq \{1, \ldots, n\}$ and $a_i \in \mathcal{A}_i$ for all $i \in I$.

In practice we do not explicitly build compositions of components. We rather interpret their semantics at runtime by evaluating enabled interactions based on current states of components. In a composition of n components $B_{i \in \{1, \cdots, n\}}$, denoted by $\gamma(B_1, \ldots, B_n)$, an action a_i can execute only as part of an interaction α such that $a_i \in \alpha$, that is, along with the execution of all other actions $a_j \in \alpha$, which corresponds to the usual notion of multiparty interaction.

Definition 2.4 (Semantics of a Composition). Given a set of components $\{B_1, \cdots, B_n\}$ and an interaction set γ. The semantics of the composition $S = \gamma(B_1, \cdots, B_n)$ w.r.t the set of interactions γ, is the LTS $(Q_g, \gamma \cup \mathbb{R}_{>0}, \to_\gamma)$ where:

- $Q_g = \mathcal{L} \times \mathcal{V}(\mathcal{X})$ is the set of global states, where $\mathcal{L} = \mathcal{L}_1 \times \cdots \times \mathcal{L}_n$ and $\mathcal{X} = \bigcup_{i=1}^{n} \mathcal{X}_i$. We write a state $q = (\ell, v)$ where $\ell = (\ell_1, \cdots, \ell_n) \in \mathcal{L}$ is a global location and $v = (v_1, \cdots, v_n) \in \mathcal{V}(\mathcal{X})$ is global clocks valuations.
- $\to_\gamma$ is the set of labeled transitions defined by the rules:
  - $(\ell, v) \xrightarrow{\alpha}_\gamma (\ell', v')$ if for $\alpha = \{a_i\}_{i \in I} \in \gamma . \forall i \in I . (\ell_i, v_i) \xrightarrow{a_i} (\ell_i, v_i)$ and $\forall i \notin I . (\ell_i, v_i) = (\ell'_i, v'_i)$ (Interaction step).

- $(\ell, v) \xrightarrow{\delta}_{\gamma} (\ell, v + \delta)$ if $\forall i \in \{1, \cdots, n\} \quad tpc_{\ell_i}(v_i + \delta)$ for $\delta \in \mathbb{R}_{>0}$ (time step).

By abuse of notation, predicates $\mathrm{at}(\ell_i)$ of individual components B_i are interpreted on states of S, being true for (ℓ, v) iff B_i is at location ℓ_i in ℓ, i.e. iff $\ell \in \mathcal{L}_1 \times \ldots \times \mathcal{L}_{i-1} \times \{\ell_i\} \times \mathcal{L}_{i+1} \times \ldots \times \mathcal{L}_n$. Similarly, clock constraints of components B_i are applied to clock valuation functions v of the composition by restricting v to clocks $\mathcal{X}_i$ of B_i. Given an interaction $\alpha \in \gamma$, these notations allow us to write $Enabled(\alpha)$ as:

$$\begin{aligned} Enabled(\alpha) &= \bigvee_{\ell=(\ell_1,\cdots,\ell_n)\in\mathcal{L}_\alpha} \mathrm{at}(\ell) \wedge guard(\alpha, \ell), \\ &= \bigvee_{(\ell_1,\cdots,\ell_n)\in\mathcal{L}_\alpha} \mathrm{at}(\ell) \wedge \bigwedge_{a_i\in\alpha} guard(a_i, \ell_i), \\ &= \bigvee_{(\ell_1,\cdots,\ell_n)\in\mathcal{L}_\alpha} \Big(\bigwedge_{i=1}^{n} \mathrm{at}(\ell_i) \wedge \bigwedge_{a_i\in\alpha} guard(a_i, \ell_i) \Big), \\ &= \bigwedge_{a_i\in\alpha} Enabled(a_i), \end{aligned}$$

where $\mathcal{L}_\alpha = \{\ell \in \mathcal{L} | \ell \xrightarrow{\alpha, g, r} \ell'\}$.

Example 2.5 (Running Example). Let us consider as a running example the composition of four components C, T_1, T_2, and R of Figure 1. Component C represents a controller that initializes, releases, and ends tasks T_1 and T_2. Tasks use the shared resource R during their execution. To implement such behavior, we consider the following interactions between C, R, and T_1: $\alpha_1 = \{init_0, init_1\}$, $\alpha_3 = \{start_0, start_1\}$, $\alpha_5 = \{take, process_1\}$, $\alpha_7 = \{end_0, free, end_1\}$, and similar interactions α_2, α_4, α_6, α_8 for task T_2, as shown by connections on Figure 1. The controller is responsible for firing the execution of each task. First, it non-deterministically initializes one of the two tasks, i.e. executes α_1 or α_2, and then releases it through interaction α_3 or α_4. Tasks perform their processing independently of the controller, after being granted an access to the shared resource (α_5 or α_6). When ended by the controller, a task releases the resource (interactions α_7 or α_8) and go back to its initial location. An example of execution sequence of the system of Figure 1 is given below, in which valuations v of clocks x, y, and z are represented as a tuples $(v(x), v(y), v(z))$:

$$\begin{aligned} &((\ell_0^1, \ell_0^2, \ell_0^3, \ell_0^4), (0, 0, 0)) \xrightarrow{5}_{\gamma} ((\ell_0^1, \ell_0^2, \ell_0^3, \ell_0^4), (5, 5, 5)) \xrightarrow{\alpha_1}_{\gamma} \\ &((\ell_1^1, \ell_1^2, \ell_0^3, \ell_0^4), (5, 5, 5)) \xrightarrow{\alpha_3}_{\gamma} ((\ell_0^1, \ell_2^2, \ell_0^3, \ell_0^4), (0, 5, 0)) \xrightarrow{2}_{\gamma} \\ &((\ell_0^1, \ell_2^2, \ell_0^3, \ell_0^4), (2, 7, 2)) \xrightarrow{\alpha_5}_{\gamma} ((\ell_0^1, \ell_3^2, \ell_0^3, \ell_1^4), (2, 7, 2)) \xrightarrow{3}_{\gamma} \\ &((\ell_0^1, \ell_3^2, \ell_0^3, \ell_1^4), (5, 10, 5)) \xrightarrow{\alpha_2}_{\gamma} ((\ell_1^1, \ell_3^2, \ell_1^3, \ell_1^4), (5, 10, 5)) \end{aligned}$$

3 LOCAL PLANNING OF INTERACTIONS

To the best of our knowledge, distributed platforms rarely offer built-in primitives for the high level coordination, required by models where different components need to synchronize together. Thus, the implementation of such models requires an execution engine (for centralized execution), or more (for a more distributed execution), responsible of coordinating components synchronizations using simpler primitives, such as point-to-point messages passing (cite bip parrallel real time) and maybe a figure.

However, the semantics presented in previous section does not distinguish between an engine decision time and the actual execution time of components. Particularly, communication delays induced by distributed platforms may introduce deadline misses during this process. Additionally, Section 2 semantics is based on the global state of the system, that is, the operational semantics rules is achieved through global states, which break the principle of distribution where interaction execution should require only knowing the state of its participating components. We introduced in (cite FM) the *weak planning semantics*, a semantics for local planning of interaction within a defined upper bounded horizon. This semantics plans interaction based only on the state of its involved components. This approach aim to differentiate the scheduling decision time of an interaction and its actual execution. In this paper we extend this work to lower bounded horizons, which effectively, represent the estimation of the target platform communication delays. In what follows, we first give some formal definitions, then we present our extension of the weak planning semantics and discuss its properties.

Preliminaries.
We define the predicate $Plannable(\alpha, \delta)$ characterizing all states from which α can be planned in δ units of time, that is, α will be *enabled* after a time progresses of δ units of time:

$$Plannable(\alpha, \delta) \implies Enabled(\alpha)[v + \delta] \quad (1)$$

Effectively, this predicate can be written as follows:

$$Plannable(\alpha, \delta) = \bigvee_{\ell\in\mathcal{L}_\alpha} \mathrm{at}(\ell) \wedge \bigwedge_{a_i\in\alpha} \Big(guard(a_i, \ell_i) + \delta \Big) \quad (2)$$

Property 1. *Let (ℓ, v) be a state of the composition S. For any interactions $\alpha, \beta \in \gamma$ such that, $(\ell, v) \xrightarrow{\beta}_{\gamma} (\ell', v')$ and $\mathrm{part}(\alpha) \cap \mathrm{part}(\beta) = \emptyset$, where $\mathrm{part}(\alpha)$ (resp. $\mathrm{part}(\beta)$) represents components participating in interaction α (resp. β), if $Plannable(\alpha, \delta)$ holds at state (ℓ, v) then it still holds at state (ℓ', v').*

This property derives from the fact that executing interactions with disjoint set of components than α does not change the states of components participating in α, that is, for $a_i \in \alpha$ we have $\ell_i = \ell'_i$ and $v_i = v'_i$.

Property 2. *Let (ℓ, v) and $(\ell, v + \delta')$, with $\delta' \in \mathbb{R}_{>0}$ be two states of the composition S. For an interaction $\alpha \in \gamma$, if $Plannable(\alpha, \delta)$ is true at state (ℓ, v) then $Plannable(\alpha, \delta - \delta')$ is true at state $(\ell, v + \delta')$ for $\delta' \leq \delta$.*

This property can be found directly by writing Equation 3 on state $(\ell, v + \delta')$.

Let $\mathcal{H}$ be a partial function $\mathcal{H} : \gamma \to \mathbb{R}_{>0} \times \mathbb{R}_{>0}$ that defines for each interaction $\alpha \in \gamma$, its respective planning horizons as an interval $[h_{\min}(\alpha), h_{\max}(\alpha)]$. We define the predicate $Plannable(\alpha)$ characterizing all states from which α can be planned w.r.t its planning horizons, that is:

$$Plannable(\alpha) \implies Enabled(\alpha)[v+\delta] \wedge \delta \in [h_{\min}(\alpha), h_{\max}(\alpha)] \quad (3)$$

Formally, it is defined as follows:

$$Plannable(\alpha) = \bigvee_{\ell\in\mathcal{L}_\alpha} \mathrm{at}(\ell) \wedge \swarrow^{h_{\max}(\alpha)}_{h_{\min}(\alpha)} \Big(\bigwedge_{a_i\in\alpha} guard(a_i, \ell_i) \Big)$$

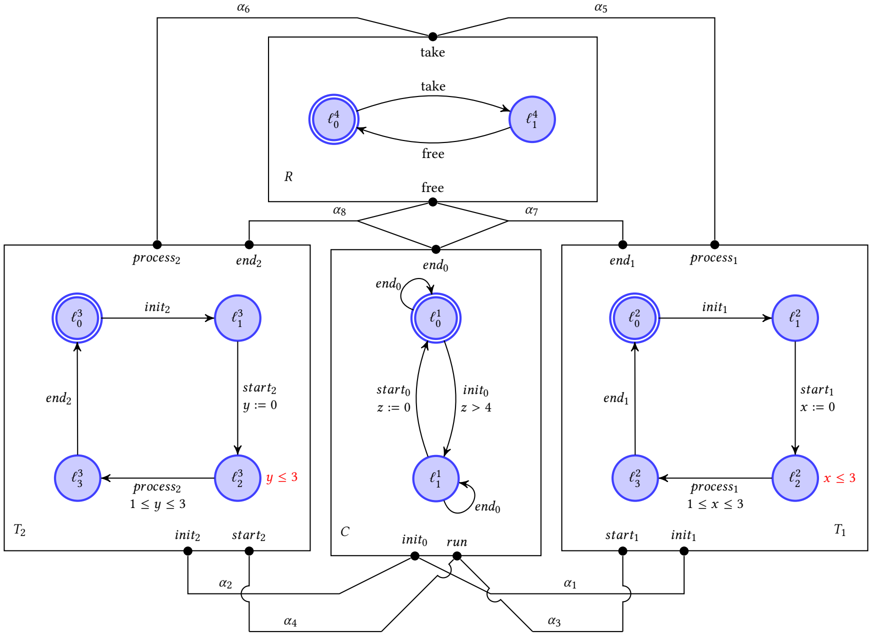

Fig. 1. Task Manager

with $\swarrow^{h_{\max}(\alpha)}_{h_{\min}(\alpha)}$ represents an adaptation of the backward operators [31] that satisfies:

$$\swarrow^{h_{\max}(\alpha)}_{h_{\min}(\alpha)} g(x) \Leftrightarrow \exists \delta \in [h_{\min}(\alpha), h_{\max}(\alpha)].g(x+\delta),$$

Property 3. *If the predicate $Plannable(\alpha, \delta)$ is true at a state (ℓ, v), then the predicate $Plannable(\alpha)$ is also true for $\delta \in [h_{\min}(\alpha), h_{\max}(\alpha)]$.*

Definition 3.1 (Plan). We say that two interactions α and β, $\alpha \neq \beta$, *conflicts* if $\text{part}(\alpha) \cap \text{part}(\beta) \neq \emptyset$, and we write $\alpha \# \beta$. A plan π is a partial function $\pi : \gamma \to \mathbb{R}_{>0}$ defining relative times for executing a subset of non conflicting interactions, i.e.:

$$\alpha \neq \alpha', \pi(\alpha) \neq \perp, \pi(\alpha') \neq \perp \implies \neg(\alpha \# \alpha').$$

We also denote by $conf(\pi)$ the set of interactions conflicting with the plan π, i.e. $conf(\pi) = \{\alpha \mid \exists \beta \# \alpha \ . \ \pi(\beta) \neq \perp\}$, and $\text{part}(\pi)$ the set of components involved in interactions planned by π, i.e. $\text{part}(\pi) = \{B_i \mid \exists \alpha \ . \ \pi(\alpha) \neq \perp \wedge B_i \in \text{part}(\alpha)\}$.

We denote by *next* π the closest relative execution time of interactions in the plan π, i.e. $\textit{next } \pi = \min\{\pi(\alpha) \mid \alpha \in \gamma \wedge \pi(\alpha) \neq \perp\} \cup \{+\infty\}$. Notice that since π stores relative times, whenever time progresses by δ the value $\pi(\alpha)$ assigned by π to an interaction α should be decreased by δ, until it reaches 0 which means that α have to execute. We write $\pi - \delta$ describing the progress of time over the plan, that is, $(\pi - \delta)(\alpha) = \pi(\alpha) - \delta$ for interactions α such that $\pi(\alpha) \neq \perp$. We also write $\pi - \alpha$ to denote the removal of interaction α from the plan π, i.e. $(\pi - \alpha)(\beta) = \pi(\beta)$ for any interaction $\beta \neq \alpha$, $(\pi - \alpha)(\alpha) = \perp$. Similarly, $\pi \cup \{\alpha \mapsto \delta\}$ assigns relative time δ to α, $\alpha \notin conf(\pi)$, into existing plan π, i.e. $(\pi \cup \{\alpha \mapsto \delta\})(\beta) = \delta$ for $\beta = \alpha$, $(\pi \cup \{\alpha \mapsto \delta\})(\beta) = \pi(\beta)$ otherwise. Finally, the plan π such that $\pi(\alpha) = \perp$ for all interactions $\alpha \in \gamma$ is denoted by $\emptyset$.

We define below the semantics for planning each interaction $\alpha \in \gamma$ with δ-horizon within $[h_{\min}(\alpha), h_{\max}(\alpha)]$.

Definition 3.2 (Planning Semantics). Given a set of components $\{B_1, \cdots, B_n\}$ and an interaction set γ, we define the planning semantics of the composition $\gamma(B_1, \cdots, B_n)$, as the labeled transition system $S_p = (Q_p, \gamma \cup \mathbb{R}_{>0} \cup \{\mathbf{plan}\}, \rightsquigarrow_\gamma)$ where:

- $Q_p = \mathcal{L} \times \mathcal{V}(\mathcal{X}) \times \Pi$, where $\mathcal{L}$ is the set of global location, $\mathcal{V}(\mathcal{X})$ is the set of global clocks valuations, and Π is the set of plans.
- **plan** defines the action of planning interactions
- $\rightsquigarrow_\gamma$ is the set of labeled transitions defined by the rules:
  - Plan: $\alpha \in \gamma, \delta \in [h_{\min}(\alpha), h_{\max}(\alpha)]$

$$\frac{\alpha \notin conf(\pi) \wedge Plannable(\alpha, \delta)}{(\ell, v, \pi) \overset{\mathbf{plan}(\alpha,\delta)}{\rightsquigarrow}_\gamma (\ell, v, \pi \cup \{\alpha \mapsto \delta\}).}$$

  - Exec: $\alpha \in \gamma$

$$\frac{\pi(\alpha) = 0}{(\ell, v, \pi) \overset{\alpha}{\rightsquigarrow}_\gamma (\ell', v', \pi - \alpha)\}}$$

  - Time Progress: $\delta \in \mathbb{R}_{>0}$

$$\frac{\delta \leq \text{next } \pi \wedge tpc_i(v_i + \delta)_{i \in \{1, \cdots, n\}}}{(\ell, v, \pi) \overset{\delta}{\rightsquigarrow}_\gamma (\ell, v + \delta, \pi - \delta)}$$

Example 3.3. Let us consider the following execution sequence for example of Figure 1 under the planning semantics rules, and for horizons values within 1 and 3 TU for all interactions.

$$((\ell_0^1, \ell_0^2, \ell_0^3, \ell_0^4), (0,0,0), \emptyset) \overset{\mathbf{plan}(\alpha_1,1)}{\rightsquigarrow}_\gamma \cdots \overset{1}{\rightsquigarrow}_\gamma \cdots \overset{\alpha_1}{\rightsquigarrow}_\gamma \cdots$$
$$\overset{\mathbf{plan}(\alpha_3,1)}{\rightsquigarrow}_\gamma \cdots \overset{1}{\rightsquigarrow}_\gamma \cdots \overset{\alpha_3}{\rightsquigarrow}_\gamma \cdots \overset{\mathbf{plan}(\alpha_2,1)}{\rightsquigarrow}_\gamma \cdots \overset{\mathbf{plan}(\alpha_5,3)}{\rightsquigarrow}_\gamma \cdots$$
$$\overset{1}{\rightsquigarrow}_\gamma \cdots \overset{\alpha_2}{\rightsquigarrow}_\gamma \cdots \overset{\mathbf{plan}(\alpha_4,1)}{\rightsquigarrow}_\gamma \cdots \overset{1}{\rightsquigarrow}_\gamma \cdots \overset{\alpha_4}{\rightsquigarrow}_\gamma \cdots \overset{1}{\rightsquigarrow}_\gamma \cdots$$
$$\overset{\alpha_5}{\rightsquigarrow}_\gamma \cdots \overset{\mathbf{plan}(\alpha_7,3)}{\rightsquigarrow}_\gamma \cdots \overset{2}{\rightsquigarrow}_\gamma ((\ell_0^1, \ell_3^2, \ell_2^3, \ell_1^4), (3,5,3), \{\alpha_7 \mapsto 1\})$$

This execution sequence represents a path that alternates plan actions, time steps and execution of some interactions. We can see that for interaction α_7 which is planned 3 units of time ahead, the system cannot reach the state from which it can be executed since there is a time progress expiration in component T_2. This execution sequence shows that a given system can deadlock under the planning semantics, even if it is deadlock-free under the global semantics rules.

3.1 Relation between Global and Weak Planning Semantics

We use weak simulation to compare the model under the global state semantics rules and the one under the planning semantics rules by considering **plan**-transitions unobservable. As explained in Example 3.3, the planning semantics does not preserve the deadlock freedom property of our system. Nevertheless, the following proves weak simulation relations between the two semantics.

Lemma 3.4. *For all the reachable states (ℓ, v, π) of the planning semantics, and $\forall \alpha \in \pi$, the predicate Plannable$(\alpha, \pi(\alpha))$ is true.*

Let $LTS_g = (\mathrm{Q}_g, \gamma \cup \mathbb{R}_{>0}, \rightarrow_\gamma)$ (resp. $LTS_p = (\mathrm{Q}_p, \gamma \cup \mathbb{R}_{>0} \cup \{\mathbf{plan}\}, \rightsquigarrow_\gamma)$) the labeled transition system characterizing the global (resp. planning) semantics.

Proposition 3.5.

R1 $\forall \delta \in \mathbb{R}_{>0}.(\ell, v, \pi) \overset{\delta}{\rightsquigarrow}_\gamma (\ell', v', \pi') \Rightarrow (\ell, v) \xrightarrow{\delta}_\gamma (\ell', v')$

R2 $\forall \alpha \in \gamma.(\ell, v, \pi) \overset{\alpha}{\rightsquigarrow}_\gamma (\ell', v', \pi') \Rightarrow (\ell, v) \xrightarrow{\alpha}_\gamma (\ell', v')$

It is straightforward that R1 is a consequence of the definition of time progress in the planning semantics. For relation R2, using Definition 3.1, we can deduce that:

$$(\ell, v, \pi) \overset{\alpha}{\rightsquigarrow}_\gamma (\ell', v', \pi') \Rightarrow \pi(\alpha) = 0,$$

By Lemma 3.4, this implies that *Plannable*$(\alpha, 0)$ is *true* at state (ℓ, v, π), meaning that *Enabled*(α) is also *true*, which allows to infer R2.

Corollary 3.6. *If a state $(\ell, v, \pi) \in Reach(LTS_p)$, then $(\ell, v) \in Reach(LTS_g)$.*

Definition 3.7 (Weak Simulation). A weak simulation over $A = (\mathrm{Q}_A, \sum \cup \{\beta\}, \rightarrow_A)$ and $B = (\mathrm{Q}_B, \sum \cup \{\beta\}, \rightarrow_B)$ is a relation $R \subseteq \mathrm{Q}_A \times \mathrm{Q}_B$ such that we have: $\forall (q, r) \in R, a \in \sum .q \xrightarrow{a}_A q' \implies \exists r' : (q', r') \in R \wedge r \xrightarrow{\beta^* a \beta^*}_B r'$ and $\forall (q, r) \in R : q \xrightarrow{\beta}_A q' \implies \exists r' : (q', r') \in R \wedge r \xrightarrow{\beta^*} r'$. B simulates A, denoted by $A \sqsubseteq_R B$, means that B can do everything A does.

The definition of weak simulation is based on the unobservability of $\beta-$transitions. In our case, $\beta-$transitions corresponds to **plan**−transitions.

Corollary 3.8. *$LTS_p \sqsubseteq_{R_1} LTS_g$ with $R = \{(q, \pi); q) \in \mathrm{Q}_p \times \mathrm{Q}_g\}$.*

Corollary 3.8 corresponds to a notion of correctness of the planning semantics: any execution in the planning semantics corresponds to an execution in the global state semantics.

By contrast to (cite fm), where planning immediately, i.e. with a zero horizon, was allowed, by introducing a minimal horizon of planning interactions, the planning semantics does no longer preserve all execution sequences of the global state semantics.

As explained in the previous example, the planning semantics may introduce deadlocks as shown by the scenario presented in the Example 3.3. In the following, we present different verification methods to check the deadlock freedom of a system under the planning semantics rules.

4 PLANNING SEMANTICS CORRECTNESS

Local planning of interactions consists effectively in applying a local time step, followed by the execution of an interaction. Even if local time steps are allowed in planned components, it may be disallowed in the rest of the system. Additionally, planning horizons insert a certain latency between interactions, which can result in missing interactions deadlines in some cases. Thus, as explained earlier, the planning semantics may exhibit deadlock situations nonexistent in the system under the global state semantics, since: *(i)* it does not consider time progress conditions of components not participating in planned interactions, and *(ii)* it introduces a certain latency between interaction executions.

Since the application model does not consider delays between interactions, in case of urgency of a time progress condition, semantically, consecutive executions can happen in the global state semantics and enable thus to get ride of this urgency. Thing that does not hold in the planning semantics.

4.1 Planning Semantics Restriction

For the sake of simplicity, we assume that all interactions have the same lower bound planning horizon, that is, $\forall \alpha, \beta \in \gamma$ such that $\alpha \neq \beta$, $h_{\min}(\alpha) = h_{\min}(\beta) = h_{\min}$. In order to enforce the correctness of the planning semantics, we restrict the condition of time progress as follows:

$$\frac{\delta \leq \text{next } \pi \wedge \forall B_i \notin \text{part}(\pi).tpc_i(v_i + \delta + h_{\min})}{(\ell, v, \pi) \overset{\delta}{\rightsquigarrow}_\gamma (\ell, v + \delta, \pi - \delta)} \tag{4}$$

This restriction aims to disallow any time progress, if there is a component not participating in any interaction of the plan such that it progresses out of its allowed planning intervals.

Let $L\dot{T}S_p = (\mathrm{Q}_p, \gamma \cup \mathbb{R}_{>0} \cup \{\mathbf{plan}\}, \rightsquigarrow_\gamma)$ the labeled transition system characterizing the planning semantics under the restriction 4.

Corollary 4.1. *$L\dot{T}S_p \sqsubseteq_{R_1} LTS_g$ with $R_1 = \{(q, \pi); q) \in \mathrm{Q}_p \times \mathrm{Q}_g\}$.*

4.2 Deadlock Characterization

Let (ℓ, v, π) be a reachable state of the planning semantics. Assuming that (ℓ, v, π) is a deadlock state, this means that *(i)* time cannot progress in the system, *(ii)* no interactions can be planned or *(iii)* executed. By definition of the planning semantics we have:

R1 *(i)* $\Rightarrow \bigvee_{B_i \in S \backslash \mathrm{part}(\pi)} \bigvee_{\ell_i \in \mathcal{L}_i} \mathsf{at}(\ell_i) \wedge \mathit{urg}(\mathit{tpc}_{\ell_i}(v_i + h_{\min}))$

R2 *(ii)* $\Rightarrow \forall \alpha \in \gamma, \neg \mathit{Plannable}(\alpha) \vee \alpha \in \mathit{conf}(\pi) \cup \pi$

R3 *(iii)* $\Rightarrow \forall \alpha \in \pi, \pi(\alpha) \neq 0 \wedge \mathit{Plannable}(\alpha, \pi(\alpha))$

Using Relations R1, R2, and R3, we obtain the following necessary and sufficient condition satisfied by deadlock states:

$$\bigvee_{B_i \in S \backslash \mathrm{part}(\pi)} \underbrace{\Big(\bigvee_{\ell_i \in \mathcal{L}_i} \mathsf{at}(\ell_i) \wedge \mathit{urg}(\mathit{tpc}_{\ell_i}(v_i + h_{\min})) \Big)}_{\text{R1}}$$
$$\wedge \underbrace{\bigwedge_{\alpha \in \gamma \backslash \pi \cup \mathit{conf}(\pi)} \neg \mathit{Plannable}(\alpha)}_{\text{R2}} \tag{5}$$
$$\wedge \underbrace{\bigwedge_{\alpha \in \pi} \Big(\pi(\alpha) \neq 0 \wedge \mathit{Plannable}(\alpha, \pi(\alpha)) \Big)}_{\text{R3}}$$

Additionally, since the deadlock is caused by a component disallowing time progress, and that none of the interactions involving this component can be planned, Equation 5 becomes:

$$\bigvee_{B_i \in S \backslash \mathrm{part}(\pi)} \Big[\underbrace{\Big(\bigvee_{\ell_i \in \mathcal{L}_i} \mathsf{at}(\ell_i) \wedge \mathit{urg}(\mathit{tpc}_{\ell_i}(v_i + h_{\min})) \Big)}_{\text{A}} \wedge$$
$$\underbrace{\Big(\bigwedge_{\alpha \in \Gamma(B_i) \backslash \mathit{conf}(\pi)} \neg \mathit{Plannable}(\alpha) \vee \bigvee_{\alpha \in \Gamma(B_i) \cap \mathit{conf}(\pi)} \mathit{Plannable}(\alpha) \Big)}_{\text{B}} \Big] \tag{6}$$
$$\wedge \underbrace{\bigwedge_{\alpha \in \pi} \Big(\pi(\alpha) \neq 0 \wedge \mathit{Plannable}(\alpha, \pi(\alpha)) \Big)}_{\text{C}}$$

where $\Gamma(B_i)$ is the set of interactions involving component B_i.

Terms A of Equation 6 represents the time progress restriction as presented in 4. It expresses clocks valuations, in a component not participating in the planned interaction, $h_{\min}$ units of time before its expiration. On the other hand, term B describes the cause of the time progress restriction, that is, all interactions of $\Gamma(\ell_i)$ are not plannable either because (1) they are conflicting with the plan or, (2) because they are not in their corresponding planning states. Finally, term C depicts the existence of interactions in the plan and the fact that none of them reached yet their execution dates.

Theorem 4.2. *Let $\phi(\alpha)$ be the following predicate:*

$$\bigvee_{B_i \in S \backslash \mathrm{part}(\alpha)} \Big[\Big(\bigvee_{\ell_i \in \mathcal{L}_i} \mathsf{at}(\ell_i) \wedge \mathit{urg}(\mathit{tpc}_{\ell_i}(v_i + h_{\min})) \Big) \wedge$$
$$\Big(\bigwedge_{\beta \in \Gamma(B_i) \backslash \mathit{conf}(\alpha)} \neg \mathit{Plannable}(\beta) \vee \bigvee_{\beta \in \Gamma(B_i) \cap \mathit{conf}(\alpha)} \mathit{Plannable}(\beta) \Big) \Big] \tag{7}$$
$$\wedge \mathit{Plann\tilde{a}ble}(\alpha)$$

where $\mathit{Plann\tilde{a}ble}(\alpha)$ is the obtained predicate by replacing $h_{\min}$ by 0 in $\mathit{Plannable}(\alpha)$ and transforming the upper timing constraints of interaction α to strict guards. If a reachable state (ℓ, v, π) deadlocks then $\exists \alpha \in \gamma$ such that $\phi(\alpha)$ is satisfied.

Since reachable states of the restricted planning semantics are reachable in the global state semantics (Corollary 4.1), we can deduce that if for every interaction of γ, Equation 7 is unsatisfied, then the system under the restricted planning semantics is deadlock-free.

5 APPENDIX

Theorem 4.2. Let (ℓ, v, π) be a reachable deadlock state of the planning semantics. This means that Equation **??** is satisfied, meaning that $\exists \alpha \in \pi$ satisfying:

$$\Big(\bigwedge_{\beta \in \Gamma(B_i) \backslash \mathit{conf}(\alpha)} \neg \mathit{Plannable}(\beta) \vee \bigvee_{\beta \in \Gamma(B_i) \cap \mathit{conf}(\alpha)} \mathit{Plannable}(\beta) \Big) \wedge \pi(\alpha) \neq 0 \wedge \mathit{Plannable}(\alpha, \pi(\alpha)) \tag{8}$$

We have:

$$\pi(\alpha) \neq 0 \wedge \mathit{Plannable}(\alpha, \pi(\alpha)) \Rightarrow \mathit{Plann\tilde{a}ble}(\alpha)$$

$$\tag{9}$$

$$\bigvee_{B_i \in S \backslash \mathrm{part}(\pi)} \bigvee_{\ell_i \in \mathcal{L}_i} \mathsf{at}(\ell_i) \wedge \mathit{urg}(\mathit{tpc}_{\ell_i}(v + h_{\min})) \Rightarrow \bigvee_{B_i \in S \backslash \mathrm{part}(\alpha)} \bigvee_{\ell_i \in \mathcal{L}_i} \mathsf{at}(\ell_i) \wedge \mathit{urg} \tag{10}$$

By combining 8, 9 and 10 we obtain Equation 7, which proves Theorem 4.2

□

REFERENCES

[1] T. Abdellatif, J. Combaz, and J. Sifakis. Model-based implementation of real-time applications. In *EMSOFT*, 2010.

[2] R. Alur and D. L. Dill. A theory of timed automata. *Theor. Comput. Sci.*, 1994.

[3] L. Astefanoaei, S. B. Rayana, S. Bensalem, M. Bozga, and J. Combaz. Compositional invariant generation for timed systems. In *Tools and Algorithms for the Construction and Analysis of Systems - 20th International Conference, TACAS 2014, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2014, Grenoble, France, April 5-13, 2014. Proceedings*, pages 263–278, 2014.

[4] R. Bagrodia. A distributed algorithm to implement n-party rendevouz. In *Foundations of Software Technology and Theoretical Computer Science, Seventh Conference, Pune, India, December 17-19, 1987, Proceedings*, pages 138–152, 1987.

[5] R. Bagrodia. Process synchronization: Design and performance evaluation of distributed algorithms. *IEEE Trans. Softw. Eng.*, 15(9):1053–1065, Sept. 1989.

[6] A. Basu, M. Bozga, and J. Sifakis. Modeling heterogeneous real-time components in bip. In *Proceedings of the Fourth IEEE International Conference on Software Engineering and Formal Methods*, SEFM '06, pages 3–12, Washington, DC, USA, 2006. IEEE Computer Society.

[7] G. Behrmann, A. David, K. G. Larsen, J. Håkansson, P. Pettersson, W. Yi, and M. Hendriks. UPPAAL 4.0. In *QEST*, 2006.

[8] J. Bengtsson and W. Yi. On clock difference constraints and termination in reachability analysis of timed automata. In *ICFEM*, 2003.

[9] S. Bensalem, M. Bozga, B. Boyer, and A. Legay. Incremental generation of linear invariants for component-based systems. In *Application of Concurrency to System Design (ACSD), 2013 13th International Conference on*, pages 80–89, July 2013.

[10] S. Bensalem, M. Bozga, J. Combaz, and A. Triki. Rigorous system design flow for autonomous systems. In *Leveraging Applications of Formal Methods, Verification and Validation. Technologies for Mastering Change - 6th International Symposium, ISoLA 2014, Imperial, Corfu, Greece, October 8-11, 2014, Proceedings, Part I*, pages 184–198, 2014.

[11] D. Chabrol, V. David, C. Aussaguès, S. Louise, and F. Daumas. Deterministic distributed safety-critical real-time systems within the oasis approach. In *International Conference on Parallel and Distributed Computing Systems, PDCS 2005, November 14-16, 2005, Phoenix, AZ, USA*, pages 260–268, 2005.

[12] K. M. Chandy and J. Misra. The drinking philosopher's problem. *ACM Trans. Program. Lang. Syst.*, 6(4):632–646, 1984.

[13] K. M. Chandy and J. Misra. *Parallel Program Design: A Foundation*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1988.

[14] R. N. Charette. This car runs on code. *IEEE Spectrum*, 2009.

[15] B. Dutertre and L. de Moura. The yices smt solver. Technical report, SRI International, 2006.

[16] A. Ghosal, T. A. Henzinger, C. M. Kirsch, and M. A. A. Sanvido. Event-driven programming with logical execution times. In *Hybrid Systems: Computation and Control, 7th International Workshop, HSCC 2004, Philadelphia, PA, USA, March 25-27, 2004, Proceedings*, pages 357–371, 2004.

[17] T. A. Henzinger, C. M. Kirsch, and S. Matic. Composable code generation for distributed giotto. In *Proceedings of the 2005 ACM SIGPLAN/SIGBED Conference on Languages, Compilers, and Tools for Embedded Systems (LCTES'05), Chicago, Illinois, USA, June 15-17, 2005*, pages 21–30, 2005.

[18] T. A. Henzinger, X. Nicollin, J. Sifakis, and S. Yovine. Symbolic model checking for real-time systems. *Inf. Comput.*, 1994.

[19] Z. Jiang, M. Pajic, S. Moarref, R. Alur, and R. Mangharam. Modeling and verification of a dual chamber implantable pacemaker. In *Proceedings of the 18th International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, TACAS'12, pages 188–203, Berlin, Heidelberg, 2012. Springer-Verlag.

[20] H. Kopetz. Time-triggered real-time computing. *Annual Reviews in Control*, 27(1):3–13, 2003.

[21] H. Kopetz. An integrated architecture for dependable embedded systems. In *Proceedings of the 23rd IEEE International Symposium on Reliable Distributed Systems*, SRDS '04, pages 160–161, Washington, DC, USA, 2004. IEEE Computer Society.

[22] L. Lamport. A fast mutual exclusion algorithm. *ACM Trans. Comput. Syst.*, 5(1):1–11, Jan. 1987.

[23] M. Lindahl, P. Pettersson, and W. Yi. Formal Design and Analysis of a Gearbox Controller. *Springer International Journal of Software Tools for Technology Transfer (STTT)*, 3(3):353–368, 2001.

[24] T. Murata. Petri nets: Properties, analysis and applications. *Proceedings of the IEEE*, 77(4):541–580, Apr 1989.

[25] J. Parrow and P. Sjödin. Multiway synchronizaton verified with coupled simulation. In *CONCUR '92, Third International Conference on Concurrency Theory, Stony Brook, NY, USA, August 24-27, 1992, Proceedings*, pages 518–533, 1992.

[26] J. A. Pérez, R. Corchuelo, D. Ruiz, and M. Toro. An order-based, distributed algorithm for implementing multiparty interactions. In *Coordination Models and Languages, 5th International Conference, COORDINATION 2002, YORK, UK, April 8-11, 2002, Proceedings*, pages 250–257, 2002.

[27] J. Quilbeuf. *Distributed Implementations of Component-based Systems with Prioritized Multiparty Interactions : Application to the BIP Framework*. Theses, Université de Grenoble, Sept. 2013.

[28] S. B. Rayana, L. Astefanoaei, S. Bensalem, M. Bozga, and J. Combaz. Compositional verification for timed systems based on automatic invariant generation. *CoRR*, abs/1506.04879, 2015.

[29] M. B. Saddek Bensalem, Benoit Boyer and A. Legay. Compositional invariant generation for timed systems. Technical Report TR-2012-15, Verimag Research Report, 2012.

[30] A. Triki. *Distributed Implementation of Timed Component-based Systems*. PhD thesis, UJF, 2015.

[31] S. Tripakis. *The analysis of timed systems in practice*. PhD thesis, Joseph Fourier University, 1998.

[32] Y. Zhao, J. Liu, and E. A. Lee. A programming model for time-synchronized distributed real-time systems. In *Proceedings of the 13th IEEE Real-Time and Embedded Technology and Applications Symposium, RTAS 2007, April 3-6, 2007, Bellevue, Washington, USA*, pages 259–268, 2007.