Local Planning of Multiparty Interactions with Bounded Horizon

Mahieddine Dellabani, Jacques Combaz, Saddek Bensalem, Marius Bozga

Verimag Research Report n^o TR-2016-05

August 16, 2016

Unité Mixte de Recherche 5104 CNRS - Grenoble INP - UGA
Bâtiment IMAG
Université Grenoble Alpes
700, avenue centrale
38401 Saint Martin d'Hères
France
tel : +33 4 57 42 22 42
fax : +33 4 57 42 22 22
http://www-verimag.imag.fr/

Local Planning of Multiparty Interactions with Bounded Horizon

Mahieddine Dellabani, Jacques Combaz, Saddek Bensalem, Marius Bozga

University Grenoble Alpes, VERIMAG, F-38000 Grenoble, France
CNRS, VERIMAG, F-38000 Grenoble, Frnace
{mahieddine.dellabani,saddek.bensalem,jacques.combaz,marius.bozga}@imag.fr
http://www.verimag.fr/rsd

August 16, 2016

Abstract

Dynamic scheduling of distributed real-time systems with multiparty interactions is acknowledged to be a very hard task. For such systems, multiple schedulers are used to coordinate the parallel activities of remotely running components. In order to ensure global consistency and timing constraints, these schedulers must cope with significant communication delays while moreover, use only point-to-point message passing as communication primitive on the platform. In this paper, we investigate a formal model for such systems as compositions of timed automata subject to multiparty interactions and we propose a distributed implementation method aiming to overcome the communication delays problem through planning ahead interactions. Moreover, we develop static conditions allowing to make the planning decisions local to different schedulers and thus to decrease the overall coordination overhead. The method has been implemented and we report preliminary results on benchmarks.

Keywords: Distributed Real-Time Systems·Timed Automata·Knowledge

Reviewers:

How to cite this report:

@techreport {TR-2016-05,
title = {Local Planning of Multiparty Interactions with Bounded Horizon},
author = {Mahieddine Dellabani, Jacques Combaz, Saddek Bensalem, Marius Bozga},
institution = {{Verimag} Research Report},
number = {TR-2016-05},
year = {2016}
}

1 Introduction

Over the past few decades, real-time systems have undergone a shift from the use of single processor based hardware platforms, to large sets of interconnected and distributed computing nodes. 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.

The design and the implementation of distributed systems is acknowledged to be a very difficult task. A central question is how to efficiently coordinate parallel activities in a distributed system by means of primary communication primitives offered by the platform, such as point-to-point messages or broadcast. Considering real-time constraints brings additional complexity since any scheduling or control decision may not only impact system performance, but may also affect the satisfaction of timing constraints. 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 precomputation of optimal control strategies. For non-critical systems, the standard practice is not to rely on models for precomputing scenarios 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 leave also little room for static optimization.

In our framework systems consist of components represented as timed automata that may synchronize on particular actions to coordinate their activities. Timed automata are strictly more expressive [1] than time-deterministic systems considered in time-triggered approaches [20], [11], [16], [17]. Our framework also differs from the one proposed in [7] and [32] by considering not only binary, but also multiparty (n-ary) synchronizations, a.k.a. *interactions*, expressing the fact that a subset of components may jointly (and atomically) switch their state if given preconditions are fulfilled. Such high level coordination means are rarely part of the built-in primitives offered by distributed platforms, and thus need to be implemented using simpler ones, e.g. exchange of messages. This has been extensively studied in the untimed context [5], [4], [13], [12], [26], [25], [27], but to the best of our knowledge, it has been solved for timed systems only under the assumption of non-decreasing deadlines in [10], [30].

We contribute to this research field by proposing methods for scheduling interactions with a bounded horizon, which aims to reduce the impact of communication delays on systems execution. In particular, *(i)* we define a semantics for *planning* interactions with a bounded horizon and prove its correctness w.r.t the regular semantics, *(ii)* we present sufficient static conditions that ensure a safe planning of interactions, and *(iii)* we leverage system knowledge to reinforce those conditions.

The rest of the paper is organized as follows. In Section 2 we provide a formal definition of compositions of timed automata with respect to multiparty interactions. We also present a semantics for planning interactions with a bounded horizon. In Section 3, we study sufficient conditions for a safe planning of interactions. Thereafter, we use global knowledge of the system to refine the latest conditions for more precise results and in order to avoid unnecessary verification (Section 4). Finally, the application of previous results on various examples is presented in Section 5.

2 Timed Systems and Properties

2.1 Global State Semantics

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 1 (Component). *A* component *is a tuple* $(\mathcal{L}, \ell_0, A, T, \mathcal{X}, tpc)$ *where* $\mathcal{L}$ *is a finite set of* locations, $\ell_0 \in \mathcal{L}$ *is an* initial *location,* A *a finite set of* actions, $\mathcal{X}$ *is a finite set of* clocks, $T \subseteq \mathcal{L} \times (A \times \mathcal{C} \times 2^{\mathcal{X}}) \times \mathcal{L}$ *is a set of* transitions *labeled with an action, a guard, and a set of clocks to be reset, and*

$tpc : \mathcal{L} \to \mathcal{C}$ *assigns a* time progress condition*, tpc_ℓ, to each location, where $\mathcal{C}$ is the set of* clock constraints *defined by the following grammar:*

$$C := \textit{true} \mid x \sim ct \mid x - y \sim ct \mid C \wedge C,$$

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 $(\mathcal{L}, \ell_0, A, T, \mathcal{X}, tpc)$, we write $\ell \xrightarrow{a,g,r} \ell'$ if there exists a transition $\tau = \big(\ell, (a, g, r), \ell'\big) \in T$. We also write:

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

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 (Semantics). *A component $B = (\mathcal{L}, \ell_0, A, T, \mathcal{X}, tpc)$ defines the labeled transition system (LTS) $(\mathrm{Q}, A \cup \mathbb{R}_{>0}, \rightarrow)$ where $\mathrm{Q} \subseteq \mathcal{L} \times \mathcal{V}(\mathcal{X})$ denotes the* states *of B and $\rightarrow \subseteq \mathrm{Q} \times (A \cup \mathbb{R}_{>0}) \times Q$ 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 $tpc_\ell(v + \delta)$ (time progress).*

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 (progress) steps, that is:

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

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 [31,8,18]. 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 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:

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

where $\mathsf{at}(\ell_j)$ 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:

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

Definition 3 (Deadlock). *We say that a state (ℓ, v) of a component B deadlocks, if neither action steps nor time steps (time progress) can be done from this state. The following equation characterizes those states:*

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

In our framework, components communicate by means of *multiparty interactions*. A multiparty interaction is 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, $i = 1, \dots, n$, with disjoint sets of actions A_i, an interaction is a subset of actions $\alpha \subseteq \cup_{1 \leq i \leq n} A_i$ containing at most one action per component, i.e. $\alpha \cap A_i$ 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 $I \subseteq \{1, \dots, n\}$ and $a_i \in A_i$ for all $i \in I$.

Definition 4 (Composition). *For n components $B_i = (\mathcal{L}_i, \ell_0^i, A_i, T_i, \mathcal{X}_i, tpc_i)$, with $\mathcal{L}_j \cap \mathcal{L}_j = \emptyset$, $A_i \cap A_j = \emptyset$, and $\mathcal{X}_i \cap \mathcal{X}_j = \emptyset$ for any $i \neq j$, the* composition $\gamma(B_1, \dots, B_n)$ *w.r.t. a set of interactions γ is defined by a timed automaton $S = (\mathcal{L}, \ell_0, \gamma, T_\gamma, \mathcal{X}, tpc)$ where $\ell_0 = (\ell_0^1, \dots, \ell_0^n)$, $\mathcal{X} = \mathcal{X}_1 \cup \dots \cup \mathcal{X}_n$, $\mathcal{L} = \mathcal{L}_1 \times \dots \times \mathcal{L}_n$, $tpc = tpc_1 \wedge \dots \wedge tpc_n$ for $\ell = (\ell_1, \dots, \ell_n)$, and T_γ is such that $\ell \xrightarrow{\alpha, g, r} \ell'$ for $\alpha = \{a_i\}_{i \in I}$, $\ell = (\ell_1, \dots, \ell_n)$, and $\ell' = (\ell'_1, \dots, \ell'_n)$, if for $i \notin I$ we have $\ell'_i = \ell_i$, and for $i \in I$ we have $\ell_i \xrightarrow{a_i, g_i, r_i} \ell'_i$, and $g_\alpha = \bigwedge_{i \in I} g_i$ and $r = \bigcup_{i \in I} r_i$.*

In practice we do not explicitly build compositions of components as presented in Definition 4. 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 $\gamma(B_1, \dots, B_n)$, 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.

Property 1 (Semantics of a Composition). Given a set of components $\{B_1, \cdots, B_n\}$ and an interaction set γ. The semantics of the composite component $S = (\mathcal{L}, \ell_0, \gamma, T_\gamma, \mathcal{X}, tpc)$ w.r.t the set of interaction γ, is the labeled transition system $(\mathrm{Q}_g, \gamma \cup \mathbb{R}_{>0}, \rightarrow_\gamma)$ where:

– $\mathrm{Q}_g = \mathcal{L} \times \mathcal{V}(\mathcal{X})$ 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 a global clocks valuations.
– γ is the set of interactions
– $\rightarrow_\gamma$ is the set of labeled transitions defined by the rules:
  - Action steps:

$$\frac{\alpha = \{a_i\}_{i \in I} \in \gamma, \quad \forall i \in I.(\ell_i, v_i) \xrightarrow{a_i} (\ell'_i, v'_i), \quad \forall i \notin I.(\ell_i, v_i) = (\ell'_i, v'_i)}{(\ell, v) \xrightarrow{\alpha}_\gamma (\ell', v')}$$

  - Time (progress) steps:

$$\frac{\delta \in \mathbb{R}_{>0} \quad \forall i \in \{1, \cdots, n\} \quad tpc_i(v_i + \delta)}{(\ell, v) \xrightarrow{\delta}_\gamma (\ell, v + \delta)}$$

In what follows, we consider only deadlock-free systems w.r.t the presented semantics. By abuse of notation predicates $\mathsf{at}(\ell_i)$ of 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 $\ell \in \mathcal{L}_1 \times \dots \times \mathcal{L}_{i-1} \times \{\ell_i\} \times \mathcal{L}_{i+1} \times \dots \times \mathcal{L}_n$. Similarly, clock constraints of components B_i are applied to clock valuation functions v of S by restricting v to clocks $\mathcal{X}_i$ of B_i. Given

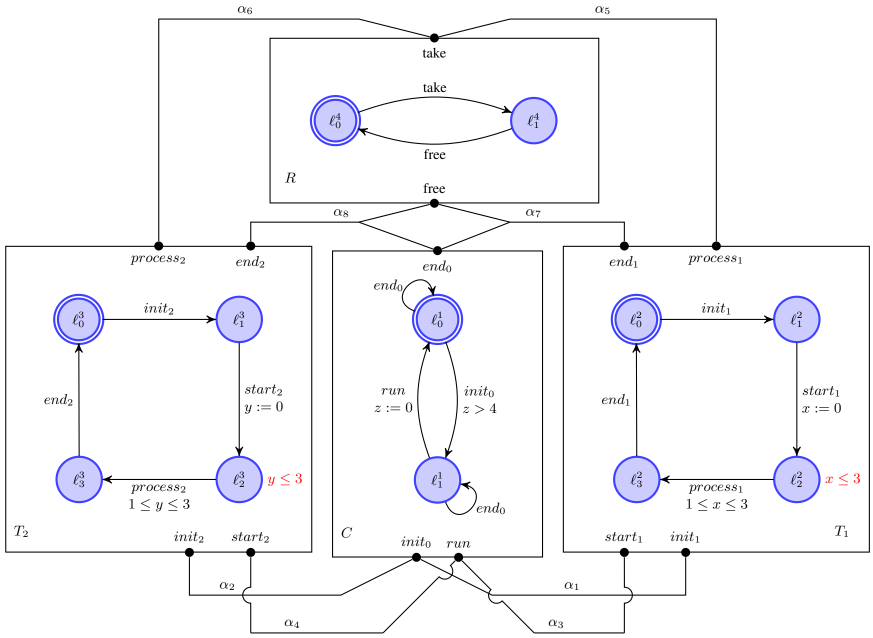

Fig. 1: Task Manager

an interaction $\alpha \in \gamma$, these notations allow us to write $\mathit{Enabled}(\alpha)$ as:

$$
\begin{aligned}
\mathit{Enabled}(\alpha) &= \bigwedge_{a_i \in \alpha} \bigvee_{(\ell_i, a_i, g_i, r_i, \ell_i') \in T_i} (\mathsf{at}(\ell_i) \wedge g_i), \\
&= \bigvee_{\ell \in \mathcal{L}_\alpha} \big(\mathsf{at}(\ell) \wedge \bigwedge_{a_i \in \alpha} \mathit{guard}(a_i, \ell_i)\big), \\
&= \bigwedge_{a_i \in \alpha} \mathit{Enabled}(a_i),
\end{aligned}
$$

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

Example 1 (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: $\alpha_1 = \{init_0, init_1\}$, $\alpha_3 = \{run, start_1\}$, $\alpha_5 = \{take, process_1\}$, $\alpha_7 = \{end_0, free, end_1\}$, 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 and 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 $(v(x), v(y), v(z))$:

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

2.2 Weak Planning Semantics

The presented semantics is based on a global state operational semantics, that is, the operational semantics rules and the computation of possible interactions between timed components is achieved through global states. Considering a distributed context, components are intrinsically concurrent and their states may be unknown until a certain time. This means that even if states of components participating in an interaction α are known, α cannot be executed in the global state semantics until the state of all components is known, which breaks the principle of distribution. Usually, components are mapped at different areas on the distributed platform in a way that better suits their interactions. In other terms, components that synchronize their actions are more likely to be next to each others. However, there are cases where several components participate in the same interaction but are mapped far from each other, which adds on communication delays to the interaction corresponding to the exchange of messages.

In order to reach an efficient scheduling, able of taking decisions ahead and using only partial (local) information, we define a different semantics based on a local planning of interactions. It aims to localize the most scheduling of interactions and alleviate the problem of communication delays through an early decision making mechanism. This is achieved by planning each interaction ahead, which means to choose an execution time within a certain horizon for each interaction, based only on the states of components involved in that interaction. Consequently, components are notified ahead through communication primitives, and will wait until the chosen execution time to perform their corresponding actions. Our approach is to define for each interaction its earliest planning date, which correspond to the maximum horizon value that ensure a safe planning of the considered interaction.

Preliminaries

We define the predicate $Enabled^{\nearrow\delta}(\alpha)$ characterizing all states from which α is *enabled* if time progresses by δ units of time, that is:

$$Enabled^{\nearrow\delta}(\alpha) = \bigvee_{\ell \in \mathcal{L}_\alpha} \big(\mathsf{at}(\ell) \wedge \bigwedge_{a_i \in \alpha} (guard(a_i, \ell_i) + \delta)\big), \tag{1}$$

Property 2. Let (ℓ, v) be a state of the composition S. For any interaction $\beta \in \gamma$ such that, $\mathrm{part}(\alpha) \cap \mathrm{part}(\beta) = \emptyset$ and $(\ell, v) \xrightarrow{\beta}_\gamma (\ell', v')$, where $\mathrm{part}(\alpha)$ (resp. $\mathrm{part}(\beta)$) represents components participating in interaction α (resp. β), if $Enabled^{\nearrow\delta}(\alpha)$ 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 3. Let (ℓ, v) and $(\ell, v + \delta')$, with $\delta' \in \mathbb{R}_{>0}$ be two states of the composition S. If $Enabled^{\nearrow\delta}(\alpha)$ is *true* at state (ℓ, v) then $Enabled^{\nearrow\delta-\delta'}(\alpha)$ is true at state $(\ell, v + \delta')$ for $\delta' \leq \delta$.

This property can be found directly by writing Equation 1 on state $(\ell, v + \delta')$.

Let $\delta_{\max}$ be a partial function $\delta_{\max} : \gamma \to \mathbb{R}_{\geq 0}$ that defines for each interaction a maximum horizon to be planned with. We define the predicate $Enabled^{\nearrow[0,\delta_{\max}(\alpha)]}(\alpha)$ characterizing all states from which α can be planned with a $\delta_{\max}(\alpha)$-horizon as follows:

$$Enabled^{\nearrow[0,\delta_{\max}(\alpha)]}(\alpha) = \bigvee_{\ell \in \mathcal{L}_\alpha} (\mathsf{at}(\ell) \wedge \swarrow^{\delta_{\max}(\alpha)} (\bigwedge_{a_i \in \alpha} guard(a_i, \ell_i))),$$

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

$$\swarrow^{\delta_{\max}(\alpha)} g(x) \Leftrightarrow \exists \delta \leq \delta_{\max}(\alpha).g(x + \delta),$$

Property 4. If the predicate $Enabled^{\nearrow\delta}(\alpha)$ is *true* at a state (ℓ, v), then the predicate $Enabled^{\nearrow[0,\delta_{\max}(\alpha)]}(\alpha)$ is also *true* for $\delta \leq \delta_{\max}(\alpha)$.

Definition 5 (Plan). *We say that two interactions α and β, $\alpha \neq \beta$,* conflicts *if* $\mathrm{part}(\alpha) \cap \mathrm{part}(\beta) \neq \emptyset$*, and we write $\alpha\#\beta$. A plan π is a partial function $\pi : \gamma \to \mathbb{R}_{\geq 0}$ 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 $\mathrm{part}(\pi)$ *the set of components involved in interactions planned by π, i.e.* $\mathrm{part}(\pi) = \{B_i \mid \exists\alpha \,.\, \pi(\alpha) \neq \perp \wedge B_i \in \mathrm{part}(\alpha)\}$.

We denote by $\min \pi$ the closest relative execution time of interactions in the plan π, i.e. $\min \pi = \min \{\pi(\alpha) \mid \alpha \in \gamma \wedge \pi(\alpha) \neq \perp\} \cup \{+\infty\}$. 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 $\beta = \alpha$, $(\pi \cup \{\alpha \mapsto \delta\})(\beta) = \pi(\alpha)$ 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 $\delta_{\max}(\alpha)$-horizon.

Definition 6 (Weak Planning Semantics). *Given a set of components $\{B_1, \cdots, B_n\}$ and an interaction set γ. We define the weak planning semantics of the composite component $S = (\mathcal{L}, \ell_0, \gamma, T_\gamma, \mathcal{X}, tpc)$, the label transition system $S_p = (\mathrm{Q}_\pi, \gamma \cup \mathbb{R}_{>0} \cup \{$***plan***$\}, \rightsquigarrow)$ where:*

- $\mathrm{Q}_\pi = \mathcal{L} \times \mathcal{V}(\mathcal{X}) \times \Pi$*, 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*
- $\rightsquigarrow$ *is the set of labeled transitions defined by the rules:*
  - *Plan:*

$$\frac{\delta \leq \delta_{\max}(\alpha), \alpha \in \gamma, \mathrm{part}(\alpha) \cap \mathrm{part}(\pi) = \emptyset \quad Enabled^{\nearrow\delta}(\alpha)}{(\ell, v, \pi) \overset{\mathbf{plan}(\alpha,\delta)}{\rightsquigarrow} (\ell, v, \pi \cup \{\alpha \mapsto \delta\}).}$$

  - *Exec:*

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

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

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

Example 2. Let us consider the following execution sequence for the example of Figure 1 under the weak planning semantics rules and for a value $\delta_{\max} = 5$ for all interactions except α_5 and α_6 that will be assigned a $\delta_{\max} = 3$:

$$
\begin{aligned}
&((\ell_0^1, \ell_0^2, \ell_0^3, \ell_0^4), (0,0,0), \emptyset) \overset{\mathbf{plan}(\alpha_1,5)}{\rightsquigarrow} ((\ell_0^1, \ell_0^2, \ell_0^3, \ell_0^4), (0,0,0), \{\alpha_1 \mapsto 5\}) \overset{5}{\rightsquigarrow} \\
&((\ell_0^1, \ell_0^2, \ell_0^3, \ell_0^4), (5,5,5), \{\alpha_1 \mapsto 0\}) \overset{\alpha_1}{\rightsquigarrow} ((\ell_1^1, \ell_1^2, \ell_0^3, \ell_0^4), (5,5,5), \emptyset) \overset{\mathbf{plan}(\alpha_3,2)}{\rightsquigarrow} \\
&((\ell_1^1, \ell_1^2, \ell_0^3, \ell_0^4), (5,5,5), \{\alpha_3 \mapsto 2\}) \overset{2}{\rightsquigarrow} ((\ell_1^1, \ell_1^2, \ell_0^3, \ell_0^4), (7,7,7), \{\alpha_3 \mapsto 0\}) \overset{\alpha_3}{\rightsquigarrow} \\
&((\ell_0^1, \ell_2^2, \ell_0^3, \ell_0^4), (0,7,0), \emptyset) \overset{\mathbf{plan}(\alpha_5,2)}{\rightsquigarrow} ((\ell_0^1, \ell_2^2, \ell_0^3, \ell_0^4), (0,7,0), \{\alpha_5 \mapsto 2\}) \overset{2}{\rightsquigarrow} \\
&((\ell_0^1, \ell_2^2, \ell_0^3, \ell_0^4), (2,9,2), \{\alpha_5 \mapsto 0\}) \overset{\alpha_5}{\rightsquigarrow} ((\ell_0^1, \ell_3^2, \ell_0^3, \ell_1^4), (2,9,2), \emptyset) \overset{\mathbf{plan}(\alpha_2,3)}{\rightsquigarrow} \\
&((\ell_0^1, \ell_3^2, \ell_0^3, \ell_1^4), (2,9,2), \{\alpha_2 \mapsto 3\}) \overset{3}{\rightsquigarrow} ((\ell_0^1, \ell_3^2, \ell_0^3, \ell_1^4), (5,12,5), \{\alpha_2 \mapsto 0\}) \overset{\alpha_2}{\rightsquigarrow} \\
&((\ell_1^1, \ell_3^2, \ell_1^3, \ell_1^4), (5,12,5), \emptyset) \overset{\mathbf{plan}(\alpha_4,0)}{\rightsquigarrow} ((\ell_1^1, \ell_3^2, \ell_1^3, \ell_1^4), (5,12,5)\{\alpha_4 \mapsto 0\}) \overset{\alpha_4}{\rightsquigarrow} \\
&((\ell_0^1, \ell_3^2, \ell_2^3, \ell_1^4), (5,0,0), \emptyset) \overset{\mathbf{plan}(\alpha_7,4)}{\rightsquigarrow} ((\ell_0^1, \ell_3^2, \ell_2^3, \ell_1^4), (5,0,0), \{\alpha_7 \mapsto 4\}) \overset{3}{\rightsquigarrow} \\
&((\ell_0^1, \ell_3^2, \ell_2^3, \ell_1^4), (8,3,3), \{\alpha_7 \mapsto 1\})
\end{aligned}
$$

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 4 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 after 3 time units from planning this interaction. This means that local planning of interactions doesn't always allow the progress of time and may thus introduce deadlocks even if the system under the global semantics rules is deadlock-free.

2.3 Relation between Global and Weak Planning Semantics

We use weak simulation to compare the model under the global semantics rules and the one under the weak planning semantics rules by considering **plan**-transitions unobservable. As explained in Example 2, the weak planning semantics does not preserve the deadlock property of our system. Nevertheless, the following proves weak simulation relations between the two semantics.

Theorem 1. *For all the reachable states (ℓ, v, π) of the weak planning semantics, and $\forall \alpha \in \pi$, the predicate $Enabled^{\nearrow \pi(\alpha)}(\alpha)$ is true.*

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

Proposition 1.

Relation 1 $\forall \delta \in \mathbb{R}_{>0}.(\ell, v, \pi) \overset{\delta}{\rightsquigarrow} (\ell', v', \pi') \Rightarrow (\ell, v) \xrightarrow{\delta}_\gamma (\ell', v')$

Relation 2 $\forall \alpha \in \gamma.(\ell, v, \pi) \overset{\alpha}{\rightsquigarrow} (\ell', v', \pi') \Rightarrow (\ell, v) \xrightarrow{\alpha}_\gamma (\ell', v')$

It is straightforward that Relation 1 is a consequence of the definition of time progress in the weak planning semantics. On the other hand, we can deduce using the definition of the weak planning semantics that:

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

By Theorem 1, this implies that $Enabled^{\nearrow 0}(\alpha)$ is *true* at state (ℓ, v, π), meaning that $Enabled(\alpha)$ is also *true*, which allows to infer Relation 2.

Corollary 1. *If a state $(\ell, v, \pi) \in Reach(S_p)$, then $(\ell, v) \in Reach(S_g)$.*

Definition 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.*

Corollary 2. $S_p \sqsubseteq_{R_1} S_g$ *with* $R_1 = \{(q, \pi); q) \in \mathrm{Q}_p \times \mathrm{Q}_g\}$.

Corollary 2 corresponds to a notion of correctness of the weak planning semantics: any execution in weak planning semantics corresponds to an execution in the global state semantics.

Theorem 2. *$S_g \sqsubseteq_{R_2} S_p$ with $R_2 = \{(q; (q, \pi)) \in \mathrm{Q}_g \times \mathrm{Q}_p | \pi = \emptyset\}$.*

Theorem 2 states that the weak planning semantics preserves all execution sequences of the global state semantics. They are obtained using immediate planning, i.e. plans π such that $\pi(\alpha) = 0$ or $\pi(\alpha) = \perp$. The weak planning semantics aims to localize the scheduling of interactions which reduces the impact of communication delays in the system, and thus is more suitable for distributed real-time systems than the global state semantics. It does not restrict the behavior of the global state semantics (see Theorem 2), and it executes ony sequences allowed by the global state semantics (see Corollary 2). However, it may introduce deadlocks as shown by the scenario presented in Example 2. In the following, we present sufficient conditions for deadlock-free planning of interactions.

3 Deadlock-free Planning

As explained in Example 2, local planning of interactions can introduce deadlocks in the system since it does not consider time progress conditions of components not participating in the planned interactions. Effectively, the weak planning semantics ensures that time can progress until the chosen execution date only w.r.t timing constraints of participating components, but such progress may be disallowed by the rest of the system leading to deadlock states. In this section, we provide sufficient conditions for having deadlock-free planning.

Planning an interaction α implies not only blocking components participating in α until α executes, but also preventing the system from planning interactions involving these components, that is, interactions of $conf(\alpha)$. Consequently, the subset of interactions $\gamma' \subseteq \gamma$ that can be planned at a given state (ℓ, v, π) depends on the content of the plan π. It satisfies $\gamma' = \{\gamma \setminus \pi \cup conf(\pi)\}$.

By Corollary 1, a (reachable) deadlock state (ℓ, v, π) of the weak planning semantics S_p is such that (ℓ, v) is a reachable state of the global state semantics S_g. Since we assume that S_g is deadlock-free, (ℓ, v) is not a deadlock in S_g. A deadlock state (ℓ, v, π) of S_p is caused by the plan π which is restricting the execution in S_p w.r.t. S_g: interactions α of π cannot execute before $\pi(\alpha)$ time units, and interactions $\alpha \in conf(\pi)$ are blocked for (at least) max $\{\pi(\beta) \mid \beta \# \alpha\}$. Notice that due to well-formed guards, in a deadlock state (ℓ, v, π) we have necessarily $\mathsf{at}(\ell_i) \wedge urg(tpc_{\ell_i})$ for a location ℓ_i of a component $B_i \notin \mathrm{part}(\pi)$.

Theorem 3. *If a state $(\ell, v, \pi) \in Reach(S_p)$ deadlocks, the following equation is satisfied:*

$$\begin{aligned} &\underbrace{\bigwedge_{\alpha\in\pi} Enabled^{\nearrow\pi(\alpha)}(\alpha)}_{A} \quad \wedge \underbrace{\bigvee_{B_i\in S\setminus \mathrm{part}(\pi)} \bigvee_{\ell_i\in\mathcal{L}_i} \mathsf{at}(\ell_i) \wedge urg(tpc_{\ell_i})}_{B} \\ &\wedge \underbrace{\bigwedge_{\alpha\in\pi} \pi(\alpha) \neq 0 \wedge \Big(\bigvee_{\alpha\in\pi} (Enabled(\alpha) \vee \bigvee_{\alpha\in conf(\pi)} Enabled(\alpha)\Big)}_{C} \end{aligned} \tag{2}$$

Term A of Equation 2 represents an invariant of the system. On the other hand, terms B and C characterize the deadlock: Term B expresses the urgency of time progress condition in components not involved in the planned interactions, whereas, term C specifies the origin of the deadlock: it characterizes states (ℓ, v, π) of S_p for which π restricts the execution of an interaction α whereas it can be executed at (ℓ, v) in S_g. As explained above, such an interaction satisfies $\pi(\alpha) > 0$ or $\alpha \in conf(\pi)$.

It is clear that Equation 2 depends on the reachable states of the planning semantics since it explicitly depends on plans π. The following gives weaker conditions for deadlocks which are independent of the plan.

Theorem 4. *Let $\Phi(\alpha)$ be the following predicate:*

$$\widetilde{Enabled}^{\nearrow[0,\delta_{\max}(\alpha)]}(\alpha) \wedge \bigvee_{B_i\in S\setminus \mathrm{part}(\alpha)} \bigvee_{\ell_i\in\mathcal{L}_i} \mathsf{at}(\ell_i) \wedge urg(tpc_{\ell_i}) \wedge \bigvee_{\beta\in\alpha\cup conf(\alpha)} Enabled(\beta) \tag{3}$$

where $\widehat{Enabled}^{\nearrow[0,\delta_{\max}(\alpha)]}(\alpha)$ *is the result of transforming all the timing constraints of the form* $x \leq ct$ *by* $x < ct$ *in* $\swarrow^{\delta_{\max}(\alpha)} (\bigwedge_{a_i \in \alpha} guard(a_i, \ell_i))$ *of* $Enabled^{\nearrow[0,\delta_{\max}(\alpha)]}(\alpha)$*.*

If a reachable state of the system (ℓ, v, π) *deadlocks then the following is satisfied:*

$$\exists \alpha \in \gamma, \Phi(\alpha) \wedge \delta_{\max}(\alpha) \neq 0 \tag{4}$$

Let $schedule(\alpha, \delta_{\max}(\alpha))$ be the following predicate:

$$schedule(\alpha, \delta_{\max}(\alpha)) = \neg\Phi(\alpha) \vee (\delta_{\max}(\alpha) = 0)$$

Using Theorem 4 and Corollary 1, we can conclude that if for all interactions $\alpha \in \gamma$ and for all reachable states of the global state semantics S_g the predicate $schedule(\alpha, \delta_{\max}(\alpha))$ is satisfied, then the weak planning semantics is deadlock-free. Notice that given an interaction $\alpha \in \gamma$ the satisfaction of $schedule(\alpha, \delta_{\max}(\alpha))$ on $Reach(S_g)$ depends only on $\delta_{\max}(\alpha)$. Moreover, it is monotonic, that is, if it holds for $\delta_{\max}(\alpha)$ then it holds for any $\delta_{\max}(\alpha)' < \delta_{\max}(\alpha)$. This provides means for building implementations that plan interactions as soon as possible by taking for $\delta_{\max}(\alpha)$ the maximal value of δ such that $schedule(\alpha, \delta)$ holds on $Reach(S_g)$.

4 Using Knowledge to Enhance Deadlock-free Planning

In Section 3, we presented sufficient conditions that ensure a deadlock-free planning of interactions. Effectively, we use an SMT solver to check the satisfiability of those conditions on the reachable states of the planning semantics. As explained in Section 3 to prove deadlock-freedom of weak planning semantics it is sufficient to prove that for all interactions $\alpha \in \gamma$ the following formula:

$$Reach(S_g) \wedge \neg schedule(\alpha, \delta_{\max}(\alpha))$$

is unsatisfiable. In practice, we do not calculate $Reach(S_g)$ to avoid the combinatorial explosion problem inherent to composition of timed automata. Instead, we use over-approximations of the latter which enable us to build weaker conditions of deadlock freedom. As explained in more detail below, these over-approximations take the form of invariants I (i.e. such that $Reach(S_g) \Rightarrow I$) that are used to establish deadlock freedom by checking the unsatisfiability of:

$$I \wedge \neg schedule(\alpha, \delta_{\max}(\alpha))$$

Timed Invariants

Our approach consists in pre-computing global knowledge of the system in the form of invariants that will be used to approximate $Reach(S_g)$. Locations reachable in a composition $S = \gamma(B_1, \ldots, B_n)$ are necessary combinations of reachable locations of individual components B_i, i.e, $Reach(S_g) \Rightarrow \bigwedge_{i=1}^{n} Reach(B_i)$. However, in general not all combinations are reachable since components are not fully independent as they synchronize through interaction set γ. Moreover, individual reachable states of components do not express the fact that time progresses the same way in all components.

For example, a global location may be not reachable because component locations having disjoint time progress conditions, or an interaction may be not enabled from a state because of an empty timing constraint. Such properties require additional relationships relating clocks of different components that are not available in $Reach(B_i)$ as it is is restricted to clocks of a single component.

We follow the approach of [3], [28], [29] for reinforcing individual reachable states of components with global invariants on clocks. They are induced by simultaneity of transitions execution when executing an interaction and the synchrony of time progress. To compute such invariants, additional *history* clocks are first introduced in components. History clocks are associated to actions of components and to interactions, and reset upon their execution. They do not modify the behavior since they are not involved in timing constraints. They only reveal local timing of components, relevant to the interaction layer, which allows to infer further properties referred as *history clocks inequalities* in [3], expressing the fact that the history clock of an interaction is necessary equal to history clocks of its actions after its execution and until the execution of another interaction involving these actions. By combining history clocks inequalities $\mathcal{E}(S)$ and symbolic states of components, we have:

$$\mathit{Reach}(S_g) \Rightarrow \bigwedge_{i=1}^{n} \mathit{Reach}(B_i) \wedge \quad \mathcal{E}(S_g) \tag{5}$$

Notice that for such system with multiparty interactions, other types of invariants could be used, like those of [9] that corresponds to the notion of *S-invariants* in the Petri net community [24]. Even if they are time abstracted, it is determined that they are appropriate for proving non coverage of subsets of individual locations.

Example 3. We illustrate the application of (5) for a safe planning of interactions by considering again example of Figure 1. It can be shown that for planning interaction α_5 (resp. α_6), we can exclude locations configurations including location ℓ_2^3 (resp. ℓ_2^2). In the following, we prove how such configurations can be excluded using history clocks inequalities.

Since action run of C is synchronized with either $start_1$ of T_1 or $start_2$ of T_2, and since history clocks h_a of an action a is reset whenever a is executed, by [3] the history clock inequalities for run are:

$$(h_{run} = h_{start_1} < h_{start_2} - 4) \vee (h_{run} = h_{start_2} < h_{start_1} - 4). \tag{6}$$

Equation (6) states that h_{run} is equal to the history clock corresponding to the last synchronization, i.e. either h_{start_1} or h_{start_2}, and is lower than history clocks of previous synchronizations. Value 4 in (6) is obtained considering *separation constraints* computed from symbolic states of components [3]: two occurrences of run are separated by at least 4 time units because of timing constraints of C, and so do occurrences of $start_1$ or $start_2$ which can only execute jointly with run. To relate history clocks with components clocks, we simply include history clocks when computing symbolic states of components (i.e. $\mathit{Reach}(B_i)$ for components), which is used to establish here that $x = h_{start_1}$ and $y = h_{start_2}$. That is, with (6) we obtain $x < y - 4$ or $y < x - 4$.

By definition of $\mathit{Enabled}$ we have $\mathit{Enabled}(\alpha_6) = \mathsf{at}(\ell_2^2) \wedge (1 \leq x \leq 3)$. Similarly, $\mathit{Enabled}(\alpha_6) = \mathsf{at}(\ell_2^3) \wedge (1 \leq y \leq 3)$. This proves that components T_1 and T_2 can never be at locations ℓ_2^3 and ℓ_2^2 at the same time. Thus, while checking for interaction α_5 (resp. α_6) that $\bigwedge_{i=1}^{n} \mathit{Reach}(B_i) \wedge \mathcal{E}(S_g) \wedge \neg \mathit{schedule}(\alpha, \delta_{\max}(\alpha))$ is unsatisfiable, this case will be excluded using history clock inequalities.

5 Implementation and Experiments

The presented method has been implemented as a middleend filter of the BIP compiler. BIP [6] is a highly expressive, component-based framework with rigorous semantics that allows the construction of complex, hierachically structured models from single components characterized by their behavior. The method input consists of real-time BIP model and a file containing an approximation of the reachable states of components combined with history clock inequalities as explained in Section 4. The latter is generated using the RTD-Finder tool, a verification tool for real-time component based systems modeled in the RT-BIP language. Our filter generates for each interaction of the input model a Yices [15] file containing system invariants together with the condition for planning the considered interaction, that is, $\neg \mathit{schedule}(\alpha, \delta_{\max}(\alpha))$. Thereafter, Yices checks the satisfiability of $\bigwedge_{i=1}^{n} \mathit{Reach}(B_i) \wedge \quad \mathcal{E}(S_g) \quad \wedge \quad \neg \mathit{schedule}(\alpha, \delta_{\max}(\alpha))$. We also define $\delta_{\max}(\alpha)$ as free variable. If this condition is unsatisfiable, then planning interactions α is safe and unbounded that is, $\delta_{\max} = +\infty$. Otherwise, Yices generates a counter-example. This counter-example can be used to find the maximal value of $\delta_{\max}(\alpha)$ satisfying the above condition using a binary search algorithm.

We ran our experiments on three other models besides of the model presented in Figure 1: Pacemaker [19], Fischer [22], and Gear controller [23]. We developed an implementation of these models in RT-BIP. The following tables show the result of our experiments. Table 1 gives a detailed result of the experiments ran on the Task Manager model 1. It summarizes, for each interaction, its *Conflicting Interactions* and the potential time progress conditions that may expire while planning it (column tpc). The last column, $\delta_{\max}(\alpha)$, details the maximum horizon for planning interaction α. Notice that the symmetry of the model allows to perform the verification on interactions $\alpha_1, \alpha_3, \alpha_5$, and α_7 and deduce the results for the other interactions. Table 2 depicts the results of our experiments on different models. For each model, it summarizes the number of interactions that can be safely planned with an unbounded horizon ($\delta_{\max} = \infty$). It also gives the number of interactions that need to be planned immediately ($\delta_{\max} = 0$).

Table 1: Detailed Results of the Task Manager Experiments

Interaction	Conflicting Interactions	*tpc*	$\delta_{\max}(\alpha)$
α_1	$\alpha_2, \alpha_4, \alpha_8$	ℓ_2^3	∞
α_3	$\alpha_2, \alpha_4, \alpha_8$	ℓ_2^3	∞
α_5	α_6, α_8	ℓ_2^3	∞
α_7	α_2, α_4	ℓ_2^3	0

Table 2: Results of Experiments

Model	Number of Interactions		
	$\delta_{\max} = 0$	$\delta_{\max} = \infty$	total
Task Manager	2	6	8
Pacemaker	0	6	6
Gear	0	17	17
Fischer	0	10	10

6 Conclusion and Future Work

We presented a method for enhancing the scheduling of real-time systems in a distributed context on models described using multiparty interactions. The proposed approach defines sufficient conditions ensuring a deadlock-free local planning of interactions with a certain horizon. Moreover, it is determined that those conditions are interaction dependent, in other terms, this means that changing the planning horizon of an interaction does not affect the planning of other interactions. A key innovative idea is the use of global knowledge in addition to local components informations to enhance the local scheduling of interactions. The computed knowledge captures not only the way components synchronize through interactions, but it also consider the separation constraints between those interactions and express explicitly the synchrony of time progress.

There are many open problems to be investigated such as: *(i)* considering more conditions when planning an interaction based on the state of components involved in this interaction, and *(ii)* defining a lower bound for planning interaction. The latter represents an important point meaning that, if planning interactions can be ensured for a lower bound, that effectively represents the communication delays of the target platform, than all the problems induced by those delays, such as global consistency and performance dropping will be solved.

Proof (Theorem 1). We will use proof by induction to prove theorem 1.

1. At the initial state of the system (ℓ_0, v_0, π_0), we have: $\pi_0 = \emptyset$. We deduce then that the theorem holds at the initial state of the system.
2. Assuming that we are at state (ℓ, v, π) where the theorem holds, let us prove that it still holds at state (ℓ', v', π') such that:
$$\forall \sigma \in \gamma \cup \mathbb{R}_{>0} \cup \{\textbf{plan}\}, (\ell, v, \pi) \overset{\sigma}{\rightsquigarrow} (\ell', v', \pi')$$
   (a) $\sigma \in \gamma$:
   We have $(\ell, v, \pi) \overset{\sigma}{\rightsquigarrow} (\ell', v', \pi') \Rightarrow \pi' = \pi - \sigma$, and we know from Definition 5 that all the interactions in a plan have disjoint set of components, that is, $\mathrm{part}(\sigma) \cap \mathrm{part}(\pi') = \emptyset$. This property allows us to infer that $\forall \alpha \in \pi'$:
$$\left.\begin{aligned} \pi(\alpha) &= \delta, \\ \pi'(\alpha) &= \delta', \end{aligned}\right\} \Rightarrow \delta = \delta',$$
   Then, using property 2, we can deduce that $\forall \alpha \in \pi', \mathit{Enabled}^{\nearrow \pi'(\alpha)}(\alpha)$ still holds at state (ℓ', v', π'). It is straightforward that for $\pi' = \emptyset$ the property holds.
   (b) $\sigma \in \mathbb{R}_{>0}$:
   We have $(\ell, v, \pi) \overset{\sigma}{\rightsquigarrow} (\ell', v', \pi') \Rightarrow \pi' = \pi - \sigma \wedge \sigma \leq \min \pi$, that is:
$$\forall \alpha \in \pi', \pi'(\alpha) = \delta' = \pi(\alpha) - \sigma \wedge \sigma \leq \pi(\alpha),$$
   Notice that:
$$\forall \alpha \in \pi', \mathit{Enabled}^{\nearrow \pi'(\alpha)}(\alpha) = \mathit{Enabled}^{\nearrow \pi(\alpha) - \sigma}(\alpha),$$
   Then, we deduce using property 3, that $\forall \alpha \in \pi', \mathit{Enabled}^{\nearrow \pi'(\alpha)}(\alpha)$ still holds at state (ℓ', v', π'). It is straightforward that for $\pi' = \emptyset$ the property holds.
   (c) $\sigma \in \{\textbf{plan}\}$:
   We have $(\ell, v, \pi) \overset{\textbf{plan}(\beta, \delta'')}{\rightsquigarrow} (\ell', v', \pi')$. By definition of planning action $\mathit{Enabled}^{\nearrow \delta''}(\beta)$ is *true* at state (ℓ', v', π'). Notice also that planning actions does not change the states (locations and clocks valuations) of components, that is, $\forall \alpha \in \pi' - \beta, \pi(\alpha) = \pi'(\alpha)$, which proves that $\mathit{Enabled}^{\nearrow \delta'}(\alpha)$ is *true* at state (ℓ', v', π').

□

Proof (Theorem 2). To prove that S_p simulates S_g, that is, $S_g \sqsubseteq_{R_2} S_p$ we need to prove that:

$$\forall (q; (q, \emptyset)) \in R_2, \forall \sigma \in \gamma \cup \mathbb{R}_{>0} \text{ such that } q \xrightarrow{\sigma}_{\gamma} q' :$$
$$\exists (q', \emptyset) \text{ such that } (q'; (q', \emptyset')) \in R_2 \text{and } (q, \emptyset) \overset{\textbf{plan}^* \sigma \textbf{plan}^*}{\rightsquigarrow} (q', \emptyset)$$

(i) $\sigma \in \gamma$:
Suppose that $(q; (q, \emptyset)) \in R_2$ and $q \xrightarrow{\sigma}_{\gamma} q'$. We have: $q \xrightarrow{\sigma}_{\gamma} q' \implies \mathit{Enabled}(\alpha) \Leftrightarrow \mathit{Enabled}^{\nearrow 0}(\alpha)$. We deduce then that, $(q, \emptyset) \overset{\textbf{plan}(\sigma, 0)\sigma}{\rightsquigarrow} (q', \emptyset)$ We conclude by remarking that $(q'; (q', \emptyset)) \in R_2$.

(ii) $\sigma \in \mathbb{R}_{>0}$:
Suppose that $(q; (q, \emptyset)) \in R_2$ and $q \xrightarrow{\sigma}_{\gamma} q'$. We have:
$q \xrightarrow{\sigma}_{\gamma} q' \implies \mathit{tpc}_i(v_i + \sigma)_{i \in \{1, \cdots, n\}}$. Thus, by definition we have $(q, \emptyset) \overset{\sigma}{\rightsquigarrow} (q', \emptyset)$. We conclude by remarking that $(q'; (q', \emptyset)) \in R_2$.

□

Proof (Theorem 3). Let (ℓ, v, π) be a reachable state of the planning semantics. It is straightforward that applying Theorem 1 to all the planned interaction, gives term A of Equation 2.

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

R1 $(i) \Rightarrow \bigvee_{i=1}^{n} \bigvee_{\ell_i \in \mathcal{L}_i} \mathsf{at}(\ell_i) \wedge \mathit{urg}(\mathit{tpc}_{\ell_i})$, with n representing the number of components in the system

R2 $(ii) \Rightarrow \forall \alpha \in \pi, \pi(\alpha) \neq 0$

R3 $(iii) \Rightarrow \forall \alpha \in \gamma \setminus \pi, \neg \mathit{Enabled}(\alpha) \vee \alpha \in \mathit{conf}(\pi)$

Notice that by putting $\delta' = \delta$ in Property 3, we can infer that $\mathit{Enabled}(\alpha)$ is satisfied at state $(\ell, v + \delta')$ and since we are assuming components with well form guards, we have the following lemma:

Lemma 1. *If $\mathit{Enabled}^{\nearrow\delta}(\alpha)$ holds at state (ℓ, v), then we have $\bigwedge_{a_i \in \alpha} \bigwedge_{\ell_i \in \mathcal{L}_i} \mathsf{at}(\ell_i) \wedge \neg \mathit{urg}(\mathit{tpc}_{\ell_i})$.*

Using this lemma, R1 can be relaxed on components participating in planned interactions, that is:

$$R1 \Rightarrow \bigvee_{B_i \in S \setminus \mathrm{part}(\pi)} \bigvee_{\ell_i \in \mathcal{L}_i} \mathsf{at}(\ell_i) \wedge \mathit{urg}(\mathit{tpc}_{\ell_i}),$$

which represent term B of Equation 2.

Using corollary 1, we know that state (ℓ, v) is reachable in the global state semantics. Moreover, we are assuming deadlock-free systems w.r.t the global state semantics, meaning that (ℓ, v) is not a deadlock state, that is, $\exists \alpha \in \gamma$ such that $(\ell, v) \xrightarrow{\alpha}_{\gamma} (\ell', v')$. By combining this with R2 and R3 we deduce that the following is satisfied:

$$\bigwedge_{\alpha \in \pi} \pi(\alpha) \neq 0 \quad \wedge \quad \big(\bigvee_{\alpha \in \pi} \mathit{Enabled}(\alpha) \vee \bigvee_{\alpha \in \mathit{conf}(\pi)} \mathit{Enabled}(\alpha) \big),$$

which gives term C and proves the theorem.

□

asd a

Proof (Theorem 4). Let (ℓ, v, π) be a rachable state of the weak planning semantics. This state is a deadlock state if Equation 2 is satisfied, meaning that $\exists \alpha \in \pi$ for which the following is satisfied:

$$\mathit{Enabled}^{\nearrow\pi(\alpha)}(\alpha) \wedge \quad \pi(\alpha) \neq 0 \quad \wedge \quad \Big(\mathit{Enabled}(\alpha) \vee \bigvee_{\beta \in \mathit{conf}(\alpha)} \mathit{Enabled}(\beta) \Big) \tag{7}$$

We have:

R1 $\pi(\alpha) \neq 0 \Rightarrow \delta_{\max}(\alpha) \neq 0$,

and using Property 4 we can infer that:

R2 $\pi(\alpha) \neq 0 \wedge \mathit{Enabled}^{\nearrow\pi(\alpha)}(\alpha) \Rightarrow \widetilde{\mathit{Enabled}^{\nearrow[0,\delta_{\max}(\alpha)]}}(\alpha)$

Thus, Equation 7 becomes:

$$(\delta_{\max}(\alpha) \neq 0) \quad \wedge \quad \widetilde{\mathit{Enabled}^{\nearrow[0,\delta_{\max}(\alpha)]}}(\alpha) \quad \wedge \bigvee_{\beta \in \alpha \cup \mathit{conf}(\alpha)} \mathit{Enabled}(\alpha) \tag{8}$$

Notice that:

$$\bigvee_{B_i \in S \setminus \mathrm{part}(\pi)} \bigvee_{\ell_i \in \mathcal{L}_i} \mathsf{at}(\ell_i) \wedge \mathit{urg}(\mathit{tpc}_{\ell_i}) \Rightarrow \bigvee_{B_i \in S \setminus \mathrm{part}(\alpha)} \bigvee_{\ell_i \in \mathcal{L}_i} \mathsf{at}(\ell_i) \wedge \mathit{urg}(\mathit{tpc}_{\ell_i}) \tag{9}$$

Finally by combining Equations 8 and 9 we obtain Equation 4, which proves the Theorem.

□

References

1. T. Abdellatif, J. Combaz, and J. Sifakis. Model-based implementation of real-time applications. In *EMSOFT*, 2010. 1
2. R. Alur and D. L. Dill. A theory of timed automata. *Theor. Comput. Sci.*, 1994. 2.1
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, 3, 3
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. 1
5. R. Bagrodia. Process synchronization: Design and performance evaluation of distributed algorithms. *IEEE Trans. Softw. Eng.*, 15(9):1053–1065, Sept. 1989. 1
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. 5
7. G. Behrmann, A. David, K. G. Larsen, J. Håkansson, P. Pettersson, W. Yi, and M. Hendriks. UPPAAL 4.0. In *QEST*, 2006. 1
8. J. Bengtsson and W. Yi. On clock difference constraints and termination in reachability analysis of timed automata. In *ICFEM*, 2003. 2.1
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. 4
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. 1
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. 1
12. K. M. Chandy and J. Misra. The drinking philosopher's problem. *ACM Trans. Program. Lang. Syst.*, 6(4):632–646, 1984. 1
13. K. M. Chandy and J. Misra. *Parallel Program Design: A Foundation*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1988. 1
14. R. N. Charette. This car runs on code. *IEEE Spectrum*, 2009. 1
15. B. Dutertre and L. de Moura. The yices smt solver. Technical report, SRI International, 2006. 5
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. 1
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. 1
18. T. A. Henzinger, X. Nicollin, J. Sifakis, and S. Yovine. Symbolic model checking for real-time systems. *Inf. Comput.*, 1994. 2.1
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. 5
20. H. Kopetz. Time-triggered real-time computing. *Annual Reviews in Control*, 27(1):3–13, 2003. 1
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. 1
22. L. Lamport. A fast mutual exclusion algorithm. *ACM Trans. Comput. Syst.*, 5(1):1–11, Jan. 1987. 5
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. 5
24. T. Murata. Petri nets: Properties, analysis and applications. *Proceedings of the IEEE*, 77(4):541–580, Apr 1989. 4
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. 1

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. 1
27. J. Quilbeuf. *Distributed Implementations of Component-based Systems with Prioritized Multiparty Interactions : Application to the BIP Framework.* Theses, Université de Grenoble, Sept. 2013. 1
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. 4
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. 4
30. A. Triki. *Distributed Implementation of Timed Component-based Systems*. PhD thesis, UJF, 2015. 1
31. S. Tripakis. *The analysis of timed systems in practice*. PhD thesis, Joseph Fourier University, 1998. 2.1, 2.2
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. 1