Noname manuscript No.
(will be inserted by the editor)

TT-BIP: Using Correct-by-Design BIP Approach for Modelling Real-Time System with Time-Triggered Paradigm

Hela Guesmi · Belgacem Ben Hedia · Simon Bliudze · Saddek Bensalem · Briag Lenabec

Received: date / Accepted: date

Abstract In order to combine advantages of Real-Time Operating Systems (RTOS) implementing the Time-Triggered (TT) execution model and model-based design frameworks, we aim at proposing a correct-by-design methodology that derives correct TT implementations from high-level models. This methodology consists of two main steps; (1) transforming the high-level model into an intermediate model which respects the TT communication principles and where all communications between components are simple send/receive interactions, and (2) transforming the obtained intermediate model into the programming language of the target platform.

In this paper, we focus on the presentation of the transformational methodology of the first step of this design flow. This methodology produces a correct-by-construction TT model by starting from a high-level model of the application software in Behaviour, Interaction, Priority (BIP). BIP is a component-based framework with formal semantics that rely on multi-party interactions for synchronizing components. Commonly in TT implementations, tasks interact with each other through a communication medium. Our methodology transforms, depending on a user-defined task mapping, high-level BIP models where communication between components is strongly synchronized, into TT model that integrates a communication medium. Thus, only inter-task communications and components participating in such interactions are concerned by the transformation process. We also provide correctness proofs of the transformation and apply it on an industrial case study.

Keywords Component-based design · time-triggered paradigm · model to model transformation · correct-by-construction transformation · formal methods

Hela Guesmi
CEA-LIST, PC 172, 91191 Gif-sur-Yvette, France
E-mail: firstname.lastname@cea.fr

Belgacem Ben Hedia
CEA-LIST, PC 172, 91191 Gif-sur-Yvette, France
E-mail: firstname.lastname@cea.fr

Simon Bliudze
INRIA Lille – Nord Europe, Parc scientifique de la Haute Borne, 40 avenue Halley 59650, Villeneuve d'Ascq, France
E-mail: simon.bliudze@inria.fr

Saddek Bensalem
Verimag / Université Grenoble Alpes, Bâtiment IMAG, 700 avenue Centrale, 38401 St Martin d'Hères, France,
E-mail: saddek.bensalem@imag.fr

Briag Lenabec
CEA-LIST, PC 172, 91191 Gif-sur-Yvette, France
E-mail: firstname.lastname@cea.fr

1 Introduction

The Time-Triggered (TT) paradigm for the design of real-time systems was introduced by Kopetz [12]. TT systems are based on a periodic clock synchronization in order to enable a TT communication and computation. Each subsystem of a TT architecture is isolated by a so-called *temporal firewall*. It consists of a shared memory element for unidirectional exchange of information between sender and receiver task components. It is the responsibility of the *TT communication system* to transport, by relying on the common global time the information from the sender firewall to the receiver firewall. The strong isolation provided by the temporal firewall is key to ensuring the determinism of task execution and, thereby, allowing the implementation of efficient scheduling policies.

Developing embedded real-time systems based on the TT paradigm is a challenging task due to the in-

creasing complexity of such systems and the necessity to manage, already in the programming model, the fine-grained temporal constraints and the low-level communication primitives imposed by the temporal firewall abstraction. Several Real-Time Operating Systems (RTOS) implement the TT execution model, such as PharOS [4] and PikeOS [11]. However, they do not provide high-level programming models that would allow the developers to think on a higher level of abstraction and to tackle the complexity of large safety-critical real-time systems. Model-based design frameworks, such as BIP [1] and SCADE [7], allow the specification, design and simulation of real-time systems. In particular, BIP—a component-based framework for the design of real-time systems—allows verification of behavioural properties, such as deadlock-freedom, and lends itself well to model transformations.

To the best of our knowledge, few connections exist between high-level component-based design frameworks, allowing reasoning about application models and verification of their functional behaviour and TT execution platforms, which guarantee temporal determinism of the system.

In this work, we propose the first step of the methodology that links between the model-based design framework BIP and TT execution platforms. This first step transforms a generic BIP model into a restricted model—called TT-BIP model. This obtained model should comply with the TT communication primitives and thereby be ready for a future transformation into the programming language of the target platform that is based on the TT paradigm. In this paper, we identify the key difficulties in defining this transformational methodology, propose exhaustive solutions to address these difficulties, provide formal transformation rules and prove that this transformation is semantics-preserving.

The rest of this paper is structured as follows. Section 2 presents the BIP framework. In Section 3, we discuss challenges of the transformation. In Section 4, we explain approach allowing to address these challenges as well as choices leading to the definition of the structure of the target TT-BIP model. In Section 5, we formally define the transformation of a high-level BIP model into a TT-BIP model. Section 7 presents the application of the proposed approach on an industrial use case. Correctness proofs of the proposed transformation are provided in the appendix 6.

2 The BIP Framework

BIP is a component framework for constructing real-time systems by superposing three layers of modelling: Behaviour, Interaction, and Priority. The Behaviour layer consists of a set of *components* defined by timed automata [3] extended with data and C functions. Transition labels of a component automaton are called *ports*. *Interactions* are sets of ports used for synchronization. Thus, the Interaction layer describes all possible synchronizations among components as a set of interactions. The third layer defines priorities among interactions, providing a mechanism for conflict resolution.

In this paper, we do not consider priorities. Thus, we only consider BIP models obtained by composing components with interactions.

2.1 Preliminary notations

Before we formally define BIP components and their semantics, we first introduce some notations. For a variable x, denote $D(x)$ its domain (i.e. the set of all values possibly taken by x). A valuation on a set of variables X is a function $v : X \to \bigcup_{x\in X} D(x)$, such that $v(x) \in D(x)$, for all $x \in X$. We denote by $\mathcal{V}(X)$ the set of all possible valuations on X and by $G_X = \mathbb{B}^{\mathcal{V}(X)}$ the set of *Boolean guards* on X.

Definition 1 (Clock constraints) Let C be a set of clocks. The associated set G_C of clock constraints CC is defined by the following grammar:

$$CC := \mathit{True} \mid \mathit{False} \mid c \sim a \mid CC \wedge CC\,,$$

with $c \in C$, $\sim \in \{\leq, =, \geq\}$ and $a \in \mathbb{Z}_+$. Notice that any guard CC can be written as:

$$CC := \bigwedge_{c\in C} l_c \leq c \leq u_c, \qquad (1)$$

where $\forall c \in C,\, l_c, u_c \in \mathbb{Z}_+ \cup \{+\infty\}$.

2.2 Ports and Interfaces

Ports are particular names used for defining communication interfaces for BIP components. In BIP, we assume that every port has an associated distinct set of data variables. This set of variables is used to exchange data with other components when interactions take place.

Definition 2 (Port) A port p is defined by:

- p : The port identifier;
- X_p : The set of data variables associated with p.

A port can be made invisible to other components, and thus label only internal computational transitions. In that case, it is called *internal* port. Symmetrically, ports visible to other components are composing the communication interface of the component which is used to establish interactions with other components. These ports are called *exported* ports. We may denote *exported* ports in the remainder of this work simply by *"ports"*.

2.3 BIP component

Definition 3 A *component* is a tuple $B = (L, P, X, C, T, tpc)$, where:

- L is a finite set of locations,
- P is a finite set of ports,
- X is a finite set of local variables,
- C is a finite set of clocks,
- $T \subseteq L \times (P \times G_X \times G_C \times 2^C \times \mathcal{V}(X)^{\mathcal{V}(X)}) \times L$ is a finite set of transitions, each labelled with a port, two Boolean guards (on variables and on clocks), a set of clocks to be reset and a function updating a subset of variables of X,
- the function $tpc : L \to G_C$ assigns a *time progress condition* to each location, such that, for any $l \in L$, the constraint $tpc(l)$ is a conjunction of constraints of the form $c \leq u_c$.

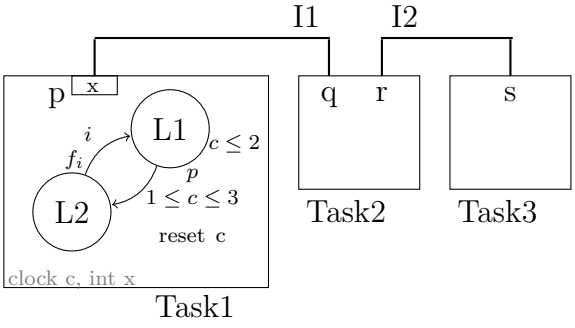


Fig. 1: RT-BIP example

Figure 1, shows a model comprising three BIP components `Task1`, `Task2` and `Task3`, composed by two binary interactions. The automaton of `Task1` is also shown in the figure. First consider `Task1` independently of the rest of the model and assume that the system reaches the state `L1` of `Task1` with $1 \leq c < 2$. Since $tpc(\texttt{L1}) = c \leq 2$, time can progress until $c = 2$ or, since the guard $1 \leq c \leq 3$ is also satisfied, transition p can be executed. If `L1` is reached with $c = 2$, the system cannot let the time progress and has to execute the transition p immediately. Finally, if $c > 2$, the time cannot progress and the system is blocked.

Definition 4 (Semantics of a component) The semantics of a component $B = (L, P, X, C, T, tpc)$ is defined as a Labelled Transition System (LTS) $(Q, P, \to)$, where $Q = L \times \mathcal{V}(X) \times \mathcal{V}(C)$ denotes the set of states of B and $\to \subseteq Q \times (P \cup \mathbb{R}_{\geqslant 0}) \times Q$ is the set of transitions defined as follows. Let (l, v_x, v_c) and (l', v'_x, v'_c) be two states, $p \in P$ and $\delta \in \mathbb{R}_{\geqslant 0}$.

- **Jump transitions**: We have $(l, v_x, v_c) \xrightarrow{p} (l', v'_x, v'_c)$ iff there exists a transition $\tau = (l, p, g_X, g_C, R, f, l') \in T$, such that $g_C(v_c) = g_X(v_x) = \mathit{True}$, $v'_x = f(v_x)$ and

$$v'_c(c) = \begin{cases} 0\,, & \text{for all } c \in R, \\ v_c(c) & \text{for all } c \in C \setminus R. \end{cases}$$

- **Delay transitions**: We have $(l, v_x, v_c) \xrightarrow{\delta} (l, v_x, v_c + \delta)$ iff $\forall \delta' \in [0, \delta]$, $tpc(l)(v_c + \delta') = \mathit{True}$, where $(v_c + \delta)(c) \stackrel{def}{=} v_c(c) + \delta$, for all $c \in C$.

A component B can execute a transition $\tau = (l, p, g_X, g_C, R, f_\tau, l')$ from a state (l, v_x, v_c) if the timing constraint g_C is met by the valuation v_c. The execution of τ corresponds to moving from control location l to l', updating variables and resetting clocks of R. Alternatively, it can wait for a duration $\delta > 0$, if the time progress condition $tpc(l)$ evaluates to *True*. This increases all the clock values by δ. Notice that execution of jump transitions is instantaneous; control location cannot change while time elapses.

2.4 Interactions

Components communicate by means of interactions. An interaction is a synchronization between transitions of a fixed subset of components. An interaction is mainly a set of ports exporting each a set of variables. An interaction can access all variables exported by its ports. Particularly, it is guarded by a predicate defined on these variables. This predicate, if evaluated to *True*, enables the interaction. This latter also defines a data transfer function which modifies the values of variables upon the execution of the interaction.

Definition 5 (Interaction) An interaction α between components $\{B_i\}_{i=1}^n$ is a triplet $(P_\alpha, G_\alpha, F_\alpha)$, where:

- P_α is a set of ports such that $|P_\alpha \cap P_i| \leqslant 1$, for all $i \in [1, n]$,
- G_α is the set of boolean guards associated to α and defined over a subset of $\bigcup_{p \in P_\alpha} X_p$.
- F_α is the set of the update functions associated to α and defined over $\bigcup_{p \in P_\alpha} X_p$.

In Definition 5, an interaction consists of one or more ports, a guard on variables associated with these ports and a data transfer function. In the remainder of this article, when no confusion is possible from the context, we may simply denote the port set of the interaction by the interaction name. Thus we may use $p \in \alpha$ instead of $p \in P_\alpha$ and $p \in \alpha_1 \cap \alpha_2$ instead of $p \in P_{\alpha_1} \cap P_{\alpha_2}$.

We denote by $comp(\alpha)$ the set of components that have ports participating in α. $comp(\alpha)$ is formally defined as:

$$comp(\alpha) = \{B_i \mid i \in [1, n], P_i \cap \alpha \neq \emptyset\}\,. \tag{2}$$

Two interactions are *conflicting* at a given state of the system if both are enabled, but it is not possible to execute both from that state (i.e., the execution of one of them disables the other). In fact, the enabledness of interactions only indirectly depends on the current state, through the enabledness of the participating ports. In systems having only the glue of interactions, two interactions α and α' may conflict only if they involve a shared component. In Figure 2a, the conflict comes from the fact that α and α' involve two ports p and q of the same component and that these two ports are labelling two transitions enabled from the same location. When reaching the location l_0, the component can execute either transition labelled by p or the one labelled by q but not both. This implies that when α and α' are enabled, only one of them should execute. Figure 2b shows a special case of conflict where interactions α and α' are sharing not only a common component but also a common port p.

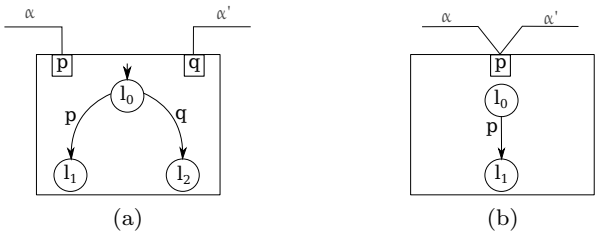


Fig. 2: Conflicting interactions

The execution of interactions in BIP framework is guaranteed by a sequential engine. This latter computes from the states of single components, the set of enabled interactions and chooses an interaction to execute among the enabled ones.

Below, we define the semantics of the model obtained by gluing BIP components with the interaction glue. This BIP semantics presented below assume atomic execution of interactions which provide sequential execution of the system.

Definition 6 (Semantics of composition with interaction model γ) Let γ be a set of interactions and let $\{B_i\}_{i=1}^n$ where $B_i = (L_i, P_i, X_i, C_i, T_i, tpc_i)$ be a set of components. The semantics of the composite component $B = \gamma(B_1, \ldots, B_n)$ is the transition system $S_\gamma = (Q, \Sigma, \rightarrow_\gamma)$ where:

- $Q = L \times \mathcal{V}(C) \times \mathcal{V}(X)$, is the set of states, with $L = L_1 \times \cdots \times L_n$ the set of global locations, $C = \bigcup_{i=1}^n C_i$ the global set of clocks and $X = \bigcup_{i=1}^n X_i$ the global set of variables. A state $q \in Q$ is of the form (l, v_c, v_x) such that $l = (l_1, \ldots, l_n)$ is the global location, $v_c = (v_{c_1}, \ldots, v_{c_n})$ is a global clocks valuation and $v_x = (v_{x_i}, \ldots, v_{x_n})$ is a global data variables valuation.
- $\Sigma = \gamma \cup \mathbb{R}_+$ is the set of labels,
- $\rightarrow_\gamma$ is the set of labelled transitions satisfying the following rules:
  - Action transitions:

$$\textsc{Inter}\ \frac{\begin{array}{c}\alpha = (\{p_i\}_{i\in I}, G_\alpha, F_\alpha) \in \gamma \\ G_\alpha(\{v_{x_i}\}_{i\in I}) \qquad \forall i \in I, (l_i, v_{c_i}, v_{x_i}) \xrightarrow{p_i}_i \\ (\{v^*_{x_i}\}_{i\in I}) = F_\alpha(\{v_{x_i}\}_{i\in I}) \\ \forall i \in I, (l_i, v_{c_i}, v^*_{x_i}) \xrightarrow{p_i}_i (l'_i, v'_{c_i}, v'_{x_i}) \\ \forall i \notin I, (l_i, v_{c_i}, v_{x_i}) = (l'_i, v'_{c_i}, v'_{x_i})\end{array}}{(l, v_c, v_x) \xrightarrow{\alpha}_\gamma (l', v'_c, v'_x)},$$

    where, in the third premise above, we use the standard shorthand $q \xrightarrow{p}$ for $\exists q' : q \xrightarrow{p} q'$.
  - Delays transitions:

$$\textsc{Del}\ \frac{\delta \in \mathbb{R}_+ \qquad \forall i \in [1,n], \forall \delta' \in [0, \delta], tpc(l_i)(v_{c_i} + \delta')}{(l, v_c, v_x) \xrightarrow{\delta}_\gamma (l, v_c + \delta, v_x)}$$

The first inference rule of Definition 6 specifies that a composite component $B = \gamma(B_1, \ldots, B_n)$ can execute an interaction $\alpha = (\{p_i\}_{i\in I}, G_\alpha, F_\alpha)$ from a global state $q = (l, v_c, v_x)$ only if (1) each port p_i is enabled in its corresponding component B_i, i.e. $q_i = (l_i, v_{c_i}, v_{x_i}) \xrightarrow{p_i}$, where q_i is the projection of the state q on the component B_i, and (2) the guard G_α defined over variables exported by ports $\{p_i\}_{i\in I}$ is evaluated to *True*. The function F is triggered by the execution of α. It modifies the variables $\{v_{x_i}\}_{i\in I}$ exported by ports $\{p_i\}_{i\in I}$. Obtained new values $\{v^*_{x_i}\}_{i\in I}$ are then processed by their respective components' transitions, which in turn can apply transformations to obtain values $\{v'_{x_i}\}_{i\in I}$. The clock valuation v'_c takes into account clocks that have been reset by their respective components' transitions. States of components which are not participating in the interaction α remain unchanged.

The second inference rule of Definition 6 states that B can execute a delay transition δ from a state $q = (l, v_c, v_x)$, only if respective time progress conditions $\{tpc_i\}_{i\in I}$ of each participating component B_i are evaluated to *True*.

3 Problem Statement

In this article, we focus on transforming BIP models in such a way that the TT communication system can be explicitly instantiated in the resulting model.

Since tasks are the building blocks of TT implementations, it would be interesting for the user to be able to specify if several components are grouped into one task of its application. Therefore, we parametrize the transformation by a user-defined task mapping which lists

different application tasks and their composing components.

In this section, we detail challenges of transforming a user-defined task mapping and a high-level BIP model based on multi-party interaction model into an equivalent model where interactions comply with the TT communication pattern. From one hand, introducing TT settings consists in (1) instantiating tasks in the derived model according to the user-defined task mapping, (2) modelling the TT communication system by introducing dedicated components and (3) restricting the synchronous multiparty inter-task interactions to simple unidirectional communications with the introduced communication components. From the other hand, the derived model is required to be observationally equivalent to the original BIP model.

In order to understand different challenges of such a transformation, consider the BIP model in Figure 3.

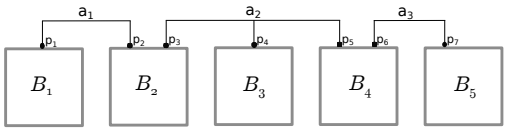


Fig. 3: High-level BIP model

In Figure 3, the model consists of five components $B_1,\ldots, B_5$ which are synchronizing through rendezvous interactions $a_1, \ldots, a_3$. In BIP framework, interactions are executed sequentially and atomically by the BIP engine. Thus, combining the need for respecting the TT settings with the need for providing the transformation correctness, requires the target model to deal with more complex issues:

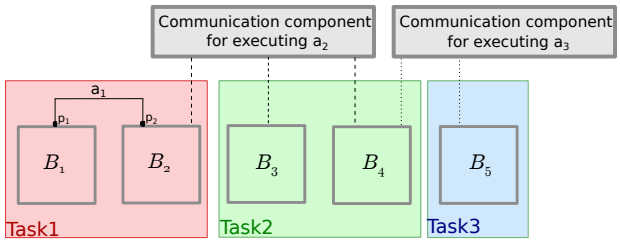
Decomposition into Tasks

Tasks (processes, threads, etc.) are building blocks of TT applications. In the design phase, designers have the choice to model a TT task using one or more BIP components. This task mapping is needed not only for defining task components but also for defining inter-task interactions that are concerned by the transformation.

For example, if we consider the task mapping displayed in Figure 4a for the model of Figure 3, then inter-task interactions are interactions a_2 and a_3. Only these two interactions have to be handled by dedicated communication components. Figure 4b shows a skeleton of the obtained model from the BIP model of Figure 3 and task mapping of Figure 4a. Dashed and dotted lines in Figure 4b display communication between tasks' components and their corresponding communication components. Details about connectors of these communications are provided by answering to the next challenge.

$$Task_1 = \{B_1, B_2\}$$
$$Task_2 = \{B_3, B_4\}$$
$$Task_3 = \{B_5\}$$

(a)



(b)

Fig. 4: Skeleton of the obtained model according to task mapping

Strong synchronization in BIP interactions Vs. asynchronous message-passing

In order to respect TT communication settings, the derived model should handle each inter-task communication through a dedicated BIP component which stands for the TT communication system. This latter can communicate with tasks only through message-passing. The challenge here is to switch from the high-level BIP model, where multi-party interactions provide component synchronization on top of data transfer, to asynchronous message-passing communications while preserving the models equivalence.

Suppose that the interaction a_2 of the example of Figure 3 allows to transfer data from component B_2 to components B_3 and B_4. Note that this interaction is atomic and allows to synchronize components B_2, B_3 and B_4. Suppose also that the dashed lines in Figure 4b present three binary connectors allowing B_2 to send data to the communication component and B_3 and B_4 to receive data from that component. Clearly, this option doesn't preserve the synchronisation between these three components ensured by the interaction a_2 in the original model since the atomicity of the original interaction is no more respected. In such a case, the communication component must be designed so that execution of interactions does not introduce behaviors that were not allowed in the initial model.

This issue is addressed by breaking the atomicity of execution of interactions. A task can execute unobservable actions to notify the communication component

about their states. If all participating components are ready, the communication component can execute the corresponding interaction.

Resolving conflicts
Suppose interaction a_2 is conflicting with interaction a_1 and/or with interaction a_3. Interaction a_2 shares with interaction a_1(resp. a_3) component B_2 (resp. B_4). Thus, a_2 can not execute concurrently with a_1 and/or with a_3. In high-level BIP model, such conflicts are resolved by the single engine. TT communication components in the derived model must ensure that execution of conflicting interactions is mutually exclusive.

4 Proposed Solution

We propose a generic framework for transforming a high-level BIP model into an equivalent model satisfying the TT settings and addressing the previously cited challenges.

The obtained model (1) expresses multiparty interactions in terms of asynchronous message passing and (2) is observationally equivalent to the initial model. The target model is structured following a three-layer architecture called TT-BIP architecture:

Layer 1
The Task Components Layer consists of a transformation of components corresponding to the behavior layer of the initial model. This layer also depends on a user-defined task mapping. A task component can interfere even in an internal computation, intra-task interaction (i.e. communication between components of the same task) or inter-task interaction (i.e. communication with other tasks). Components within a task that are concerned by the inter-task interaction or participating in an intra-task interaction that is conflicting with an inter-task interaction, only communicate with dedicated communication components.

Layer 2
The communication Layer aims at modelling the TT communication system by hosting inter-task interactions and allowing to resolve their potential conflicts by soliciting the third layer. This layer contains TT communication component (TTCC) hosting each an inter-task interaction of the original model.
We have essentially two conflict cases involving inter-task interactions; conflict between only inter-task interactions and conflict between inter-task interactions and intra-task interactions or internal computations. By dedicating a third layer for resolving conflicts, the first case of conflicts, if existing, can be directly resolved. Resolving the second conflict case, can not be resolved locally since a task has a partial observability of the system. This needs however, to host the conflicting intra-task interaction or internal computation in the communication layer in order to be resolved by requesting the third layer. Notice also that two conflicting intra-task interactions a_1 and a_2, such that a_2 is conflicting with an inter-task interaction b, need both to be handled in the communication layer. We say that a_2 is *directly* conflicting with b, while a_1 is *indirectly* conflicting with the same interaction.
Thus, this layer consists of components hosting each either an inter-task interaction or an interaction that is either *directly* or *indirectly* conflicting with another inter-task interaction. For simplifying the notation, all constituent components of the communication layer are denoted by TTCC components.

Layer 3
The Conflict Resolution Protocol (CRP) Layer resolves the conflicts requested by the communication layer. In the original model, these conflicts are resolved by the BIP engine. In order to guarantee conflicts resolution in the derived model, we reuse the same solution proposed in [10,14,15] which consists in dedicating a third layer to implement the fully centralized committee coordination algorithm presented in [5].

Cross-layer interactions are send/receive interactions, i.e. providing a unidirectional data transfer from one sender component to one or more receiver(s).

Note that tasks are building blocks of the first layer, which addresses the first challenge. The second layer allows to handle multiparty interaction of the original BIP model through dedicated communication components. And interactions between these components and task components are send/receive interactions, which addresses the second challenge. The introduction of the third layer and hosting all interactions that are conflicting with inter-task interactions in the communication layer allows to resolve the third challenge.

4.1 TT-BIP: Architecture of the Target Model

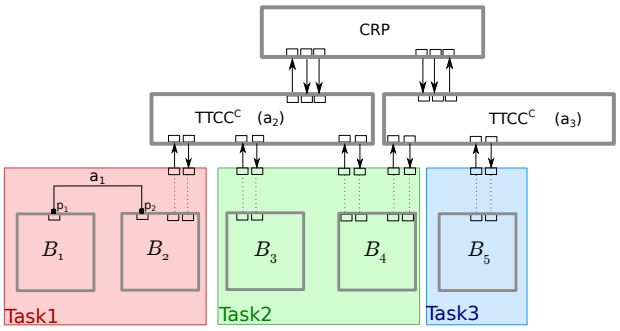
In this subsection, we present in details the TT-BIP architecture. As explained before, it imposes a structure for the target model of the transformation in order to guarantee both its compliance with the TT settings and its observational equivalence with respect to the original BIP model.

A BIP model complies with the TT-BIP architecture if it consists of three layers: Tasks layer, TTCC layer and CRP layer, organized by the following abstract grammar:

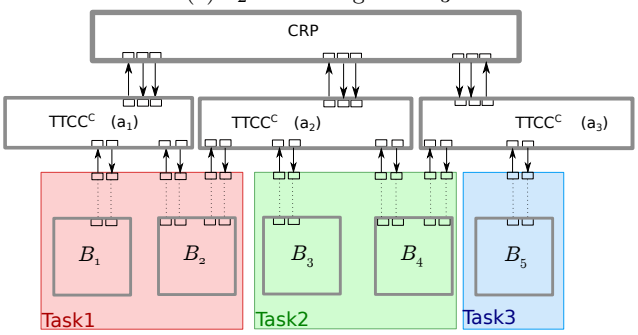
$$
\begin{aligned}
\textit{TT-BIP-Model} &::= \textit{Task}^{+} \,.\, \textit{TTCC}^{+} \,.\, \textit{CRP} \,.\, \textit{S/R-connector}^{+} \\
\textit{Task} &::= \textit{connectors}^{+} \,.\, \textit{component}^{+} \,.\, \textit{atomic-talking-component}^{+} \\
\textit{TTCC} &::= \textit{TTCC}^{NC} \mid \textit{TTCC}^{C}
\end{aligned}
$$

The TT-BIP model consists of a set of Tasks, TTCC and CRP components. A task component is a BIP component consisting of one or more components. Components within a task which interfere in inter-task interactions (via the task interface) are called *atomic-talking-components* (ATC). These latter can only communicate with a TTCC component or a component within the same task. The behavior of a TTCC component depends on whether the interaction it is hosting is conflicting or not. If the interaction is conflicting, the TTCC component is denoted by $TTCC^C$ and needs to communicate with the CRP component. Otherwise, it is denoted by $TTCC^{NC}$. Conflicts between different $TTCC^C$ components are resolved through CRP component.

Task components (resp. TTCC components) and TTCCs (resp. CRP components) communicate with each other through message-passing, i.e. send/receive interactions. Such interaction is a set of one send port and one or more receive ports. Communications between components inside a task are classic multi-party BIP interactions. Figure 5 shows an overview of the TT-BIP model derived from BIP model of Figure 3 and the task mapping displayed in Figure 4a. Notice that in Figure 5a, we assume that the interaction a_2 is conflicting only with the interaction a_3, while in Figure 5b a_2 is conflicting with both a_1 and a_3.



(a) a_2 conflicting with a_3



(b) a_2 conflicting with a_1 and a_3

Fig. 5: Overview of the TT-BIP model of the model of Figure 3

Formally, we define a TT-BIP model as follows:

Definition 7 We say that $B^{TT} = \gamma^{TT}(B_1^{TT}, \ldots, B_n^{TT})$ is a TT-BIP model iff we can partition the set of its ports into three sets P_u, P_s and P_r that are respectively the set of unary ports, send ports and receive ports, such that:

- Each interaction $\alpha \in \gamma^{TT}$ is either a send/receive interaction with $P_\alpha = s, r_1, \ldots, r_k$, $s \in P_s, r_1, \ldots, r_k \in P_r$, $G_\alpha = True$ and F_α copies variables exported by port s to variables associated with ports $r_1, \ldots, r_k$, or a unary interaction—called also *external* interaction—where $P_\alpha = p_\alpha$ with $p_\alpha \in P_u$, $G_\alpha = True$ and F_α is the identity function.
- Interactions that are relating components of the same task are classic multiparty interactions—called *internal* interaction—.
- If s is a port in P_s, then there exists one and only one send/receive interaction $\alpha \in \gamma^{TT}$ with $P_\alpha = (s, r_1, \ldots, r_k)$ and all ports $r_1, \ldots, r_k$ are receive ports. We say that $r_1, \ldots, r_k$ are receive ports of s,
- In the TT-BIP model, from the same state, an internal port can be simultaneously enabled only with another internal port. A receive port can be conflicting either with receive or send ports or both. A send port can be conflicting either with send or receive ports.
- If defined, update functions of transitions labelled by send ports do not involve data associated to the labelling port (send port).
- All transitions that are triggered by receive ports are associated with timing constraint and guards that are always default to *True*.
- If $\alpha \in \gamma^{TT}$ is a send/receive interaction such that $P_\alpha = (s, r_1, \ldots, r_k)$ and s is enabled at some global state of B^{TT}, then all its receive ports $r_1, \ldots, r_k$ are also enabled at that state.

4.2 Discussion

The proposed solution leads out to a 3-layer architecture structuring the target model of the transformation. Although our work doesn't have the same goal as transformational approaches proposed in [10,14,15], but there is some intersection between both target models' architectures. Aiming at deriving distributed implementations from high-level BIP model, these cited approaches propose an intermediate model called send/receive model. This latter is a 3-layer model consisting of components layer, schedulers layer and CRP layer.

As already mentioned —earlier in this section—, we reuse the third layer of the send/receive model (i.e. the CRP layer) since it is, so far, the unique solution to guarantee the conflicts resolution without requesting

the BIP engine. The difference between the send/receive and the TT-BIP architectures lies in the task notion introduced in the TT-BIP architecture. Thus, we build the task layer depending on a user-defined task mapping, and we construct communication components in order to handle inter-task interactions and other conflicting interactions. In the second layer of send/receive models, are introduced schedulers allowing to handle interactions between all components. Also, we introduce one component per external interaction, while a scheduler of send/receive model can handle more than one interaction.

5 Transformation of a BIP Model into a TT-BIP Model

In this section, we describe in details our technique for transforming a BIP model $B \stackrel{def}{=} \gamma(B_1, ..., B_n)$ into a TT-BIP model B^{TT} such that $B^{TT} = \gamma^{TT}(B_1^{TT}, ..., B_n^{TT}, TTCC_1, ..., TTCC_m, CRP)$.

One parameter to this transformation is the user-defined task mapping which consists in associating to each task T_k a group of components of the model B. We denote by $\mathcal{B}$ the set of components of model B. The task mapping is formally defined as follows:

Definition 8 (Task mapping) We assume, we have $K \leq n$ tasks and we denote by $\mathcal{T} = \{T_k\}_{k \in K}$ the task set, such that $\mathcal{T}$ is a partition of $\mathcal{B}$: where for all $j, k \in K$ and $j \neq k, T_j \cap T_k = \emptyset$. For all $k \in K$ we have $T_k = \{B_i\}_{i \in I_k}, I_k \subseteq K$ such that $\underset{k \in K}{\cup} I_k = K$.

The transformation process is performed in two steps as shown in Figure 6. First, depending on the given task mapping, the original model is analysed in order to define the set of components and connectors to be transformed. Then, the BIP model is transformed into a TT-BIP model where only inter-task interactions and other related conflicting interactions are replaced by TTCC components. Non conflicting intra-task interactions remain intact.

We first present details about the analysis phase in Subsection 5.1. Then, we explain how concerned components are transformed and how task components are instantiated in Subsection 5.2. Then we show how TTCC components are built in order to coordinate task components in Subsection 5.3. The behavior of the CRP component is detailed in Subsection 5.3.1. Finally, we define the cross-layer connections in Subsection 5.5.

5.1 Analysis phase

We have first to identify internal and external interactions as well as ATC components denoted respectively A_I, A_E and $\mathcal{B}^{ATC}$. These obtained sets are inputs for the transformation of components and connectors of B into B^{TT}.

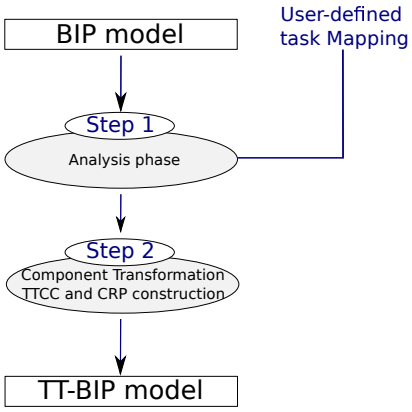


Fig. 6: A two-step transformation

External interactions
In order to be able to define the set A_E, we need first to define the set of inter-task interactions denoted A_{IT}. An interaction $a \in \gamma$ is an inter-task interaction iff at least two of its participating components belong to two different tasks. Formally,

$$A_{IT} = \left\{ \alpha \in \gamma \;\middle|\; \begin{array}{l} \exists B_1, B_2 \in comp(\alpha), T_1, T_2 \in \mathcal{T} : \\ B_1 \in T_1, B_2 \in T_2, T_1 \neq T_2 \end{array} \right\}$$

We denote intra-task interactions that are either *directly* or *indirectly* conflicting with inter-task ones by $A_{IT}^{\#}$ defined as follows:

$$A_{IT}^{\#} = \{a \in \gamma \,|\, a \notin A_{IT}, \exists \alpha \in A_{IT} : a \# \alpha\} \cup \left\{ a \in \gamma \;\middle|\; \begin{array}{l} a \notin A_{IT}, \exists b \notin A_{IT}, \\ \exists \alpha \in A_{IT} : a \neq b, a \# b, b \# \alpha \end{array} \right\}$$

And we denote by A_{IT}^{p} the set of transitions labelled by internal ports and conflicting with interactions of $A_{IT}^{\#} \cap A_{IT}$. It is defined as follows:

$$A_{IT}^{p} = \left\{ p \;\middle|\; \begin{array}{l} \forall a \in \gamma, p \notin P_a, \exists \alpha \in A_{IT} \cup A_{IT}^{\#}, : q \in P_\alpha, \\ \exists i \in [1, n], \exists l \in L_i : l \xrightarrow{p}, l \xrightarrow{q} \end{array} \right\}$$

As explained in Definition 7, A_E consists of inter-task interactions A_{IT}, intra-task interactions $A_{IT}^{\#}$ and internal transitions A_{IT}^{p} that are either *directly* or *indirectly* conflicting with inter-task ones. Thus, we have:

$$A_E = A_{IT} \cup A_{IT}^{\#} \cup A_{IT}^{p} \tag{3}$$

Internal interactions
The set A_I is defined as the set of intra-task interactions (i.e. participating components are belonging to the same task) which are neither *directly* nor *indirectly* conflicting with inter-task components:

$$A_I = \gamma \setminus A_E. \tag{4}$$

Atomic talking components
$\mathcal{B}^{ATC}$ set is the set of components in $\mathcal{B}$ that are concerned by external interactions A_E. We define:

$$\mathcal{B}^{ATC} = \{B \in \mathcal{B} | A_E \cap P_B \neq \emptyset\}, \quad (5)$$

where P_B is the set of ports of the component B.

5.2 Transformation of Task Components

We transform each ATC component $B_i \in \mathcal{B} \cap \mathcal{B}^{ATC}$ of a BIP model into a TT ATC component B_i^{TT} that is capable of communicating with $TTCC$ component(s). This transformation consists mainly in decomposing each "atomic" inter-task synchronization into send and receive actions. The synchronization between the ATC component (via the task interface) and the TTCC layer is implemented as a two-phase protocol.

First, B_i^{TT} sends communication *offers* through dedicated send ports. Then, in the second step, it waits for a notification coming from the TTCC component via a receive port. The communication *offer* contains information about the enabledness of the interaction. Each offer is associated to one of the enabled ports of B_i through which the component is ready to interact. An offer consists of a set of variables related to the corresponding enabled port. Let p be such port enabled from a location l (i.e. $l \xrightarrow{p}$). The set of variables of the corresponding offer includes variables initially exported by p since they may be read and written by the interaction. It also includes variables tc_p and tpc_l storing respectively timing constraint of transition labelled by p and enabled from l and the time progress condition of the location l. Another variable g_p is dedicated to store the evaluation of the Boolean guard of the transition labelled by p and enabled from l. The offer contains also a variable f_i storing the update function of the transition labelled by the port p. In order to be able to resolve conflicts, each offer contains the *participation count* variable nb of the component B_i^{TT}. This variable counts the number of interactions B_i^{TT} has participated in.

The notification —received after sending offers— allows the ATC component to execute the transition triggered by the enabled receive port marking the end of the interaction.

Notice that each offer —sent by a component—contains information about only one enabled interaction among the enabled interaction set. Therefore, if in the original model B, more than one interaction involving B_i are enabled, then B_i^{TT} has to send first successive offers before waiting for notification from the TTCC component executing the interaction selected after conflict resolution.

Let a location l, in B_i, from which $p_1, ..., p_n$ are enabled such that at least one of the n ports interferes in an inter-task interaction. In B_i^{TT}, we split such a location l into $n+1$ locations, namely l itself and locations $\{\perp_{p_i}^l\}_{i\in[1,n]}$ from which corresponding offers are sent (see Figure 7). Consider the case when, in the original model B_i, time is allowed to progress from location l, i.e. before executing the interaction. In order to enforce the correctness of the target model, time should be able to progress until the interaction is actually executed. Thus we associate to locations $\perp_{p_i}^l$ the time progress condition of location l originally defined in the component B_i.

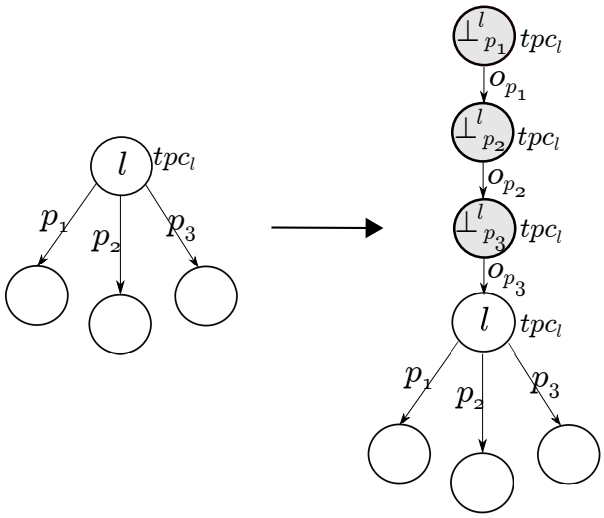


Fig. 7: Component transformation into an ATC component

5.2.1 Expressing Timing Constraints and Time Progress Conditions over a Common Global Clock

In BIP framework, each component can define its own local set of clocks. These clocks can be reset at any time and are used in definitions of timing constraints and time progress conditions.

In order to execute an external interaction $a = p_i, i \in I$, a TTCC component needs to evaluate the timing constraint of the interaction, i.e. the conjunction of timing constraints of transitions labelled by ports p_i involved in the interaction in the original model. These respective timing constraints are sent by respective ATC components to the TTCC layer within offers. In order to allow the TTCC to compute interactions between tasks components and schedule them correctly, we need to reduce the effort of keeping track of different clocks of participating components. This can be resolved by expressing timing constraints in terms of a single time scale, that is, a single global clock. Moreover, the global

time scale is a key feature of the TT paradigm targeted by the transformation.

For these two reasons, we need to translate all timing constraints and express them over the global clock. We denote by c^g, the global clock which is initialized to 0 and measures the absolute time elapsed since the system started executing, i.e. c^g is never reset.

We follow a similar approach as in [2] in order to translate selected timing constraints. Here are the different translation steps:

1. for each component $B_i \in \mathcal{B}$ and for each clock $c \in C$, we introduce a variable w_c that stores the absolute time of the last reset of c. The variable w_c is initialized to zero and updated to the absolute time (i.e. the valuation of the global clock c^g) whenever the component executes a transition resetting clock c.
2. Each atomic expressions $lb \leqslant c \leqslant ub$ involved in a timing constraint tc, is rewritten by using the global clock c^g and the variable w_c. Mainly, we have to add to the initial lower and upper bounds the last reset value w_c of the local clock c as follows:

$$lb \leqslant c \leqslant ub \equiv lb + w_c \leqslant c^g \leqslant ub + w_c \tag{6}$$

3. Similarly, we rewrite each atomic expressions $c \leqslant ub$ of time progress conditions tpc —defined on all locations from which an external interaction can be enabled—as follows:

$$c \leqslant ub \equiv c^g \leqslant ub + w_c \tag{7}$$

Notice that the value of each local clock c can be computed from the current value of the global clock c^g and the variable w_c by using the equality $c = c^g - w_c$. This allows to entirely remove clocks of components B_i, keeping only the clock c^g and variables w_c; $c \in C$.

5.2.2 Formal transformation rule

Rule 51 (Transforming ATC components) *Each ATC BIP component $B_i = (L_i, P_i, X_i, C_i, T_i, tpc_i) \in \mathcal{B}^{ATC}$ is transformed into a TT ATC component $B_i^{TT} = (L_i^{TT}, P_i^{TT}, X_i^{TT}, C_i^{TT}, T_i^{TT}, tpc_i^{TT})$ as detailed by the following rules:*

- *Each location $l \in L_i$, enabling ports $\{p_j\}_{j\in[1,n]} \subseteq P_i \cap A_E$, is split into $n+1$ locations. Obtained locations are l itself and locations $\{\perp_{p_j}^{l}\}_{j\in[1,n]}$. The time progress conditions of locations $\perp_{p_j}^{l}$ and l are equal to $tpc(l)$,*
- *Each port $p_j \in P_i \cap A_E$ such that $l \xrightarrow{p_j}$, is split into two ports; receive port p_j and send port o_{p_j}. A port $p_j \in P_i^{TT}$ exports variables $X_{p_j} \subseteq X_i$ originally exported by port $p_j \in P_i$. A port o_{p_j} exports, on top of variables $X_{p_j} \subseteq X_i$, variables tpc, tc_p, g_p, f_p and nb which are respectively the timing constraint variable, the time progress constraint variable, the Boolean guard variable, the update function variable and the participation count variable. These variables store respectively tpc of location l (i.e. $tpc(l)$) expressed on clock c^g, the timing constraint, the update function and the guard of transition enabled from l and labelled by p_j and the number of interactions the component has participated in.*
- *For each clock $c \in C_i$, we add a corresponding reset variable w_c,*
- *For each transition $\tau_{p_j} = (l, p_j, g_{\tau_{p_j}}, tc_{\tau_{p_j}}, r_{\tau_{p_j}}, f_{\tau_{p_j}}, l')$, such that $\forall j \in [1,n]$, $l \xrightarrow{p_j}$ and $p_j \in P_i \cap A_E$, we include, in T_i^{TT}, the corresponding offer transition $\tau_{o_{p_j}}$ and notification transition τ'_{p_j}. The offer transition $\tau_{o_{p_j}}$ is enabled from location $\perp_{p_j}^{l}$. Both its guard and timing constraint are True. Its update function is the identity function and it resets no clock. It reaches location $\perp_{o_{p_k}}$ if $j \neq k$ and the offer o_{p_k} is not yet sent, otherwise it reaches location l. Notification transition τ'_{p_j} is enabled from location l and reaches location l'. As in the offer transition, guard and timing constraint of the notification transition are always True. It resets the same clock set as $r_{\tau_{p_j}}$. The update function $f_{\tau'_{p_j}}$ (1) updates the clock reset variables: $\forall c \in r_{\tau_{p_j}}$, $w_c = v_c(c^g)$, where v_c is the clock valuation function, (2) increments the participation count variable nb and (3) updates variables of offers sent from next reached state.*
- *For each transition $\tau_p = (l, p, g_{\tau_p}, tc_{\tau_p}, r_{\tau_p}, f_{\tau_p}, l')$, such that $p \in P_i \setminus A_E$, we instantiate the transition τ'_p, where only the update function is changed compared to the initial transition τ_p. The update function $f_{\tau'_p}$ (1) applies the original update function f_{τ_p}, (2) updates the clock reset variables: $\forall c \in r_{\tau_{p_j}}$, $w_c = v_c(c^g)$, where v_c is the clock valuation function, (3) increments the participation count variable nb and (4) updates variables of offers sent from next reached state.*
- *In order to update variables of offers that will be sent from its reached location l', a transition needs to execute the following functions:*
  - *$tpc := tpc(l')^{c^g}$, where $tpc(l')^{c^g}$ corresponds to expressing the tpc of l' over the global clock c^g following (7),*
  - *$\forall p \in P_i \cap A_E$, such that $\exists \tau_p = (l', p, g_{\tau_p}, tc_{\tau_p}, r_{\tau_p}, f_{\tau_p}, l'') \in T_i$, $tc_p := tc_{\tau_p}^{c^g}$, $g_p = g_{\tau_p}$ and $f_p := f_{\tau_p}$, where $tc_{\tau_p}^{c^g}$ corresponds to expressing the timing constraint of τ_p over the global clock c^g following (6) and g_{τ_p} is the guard evaluation.*

After applying Rule 51, we can formally define the obtained component in function of the original one.

Definition 9 Formally, B_i^{TT} is obtained from B_i as follows:

- $L_i^{TT} = L_i \cup L_\perp$, where $L_\perp = \{\perp_p^l \mid \exists l \in L_i, \exists \tau = (l, p, g, tc, r, f, l') \in T_i, p \in P_i \cap A_E\}$,
- $P_i^{TT} = P_i \cup P_o$, where $P_o = \{o_p | p \in P_i \cap A_E\}$. Each port o_p exports the set of variables $X_{o_p}^{TT} = X_p \cup \{tpc, tc_p, g_p, f_p, nb\}$. For all ports in $p \in P_i \cap A_E$, we have $X_p^{TT} = X_p$. For all ports $p \in P_i \setminus A_I$, we have $X_p^{TT} = X_p$. ,
- $X_i^{TT} = X_i \cup \{tpc\} \cup \{tc_p, g_p, f_p\}_{p \in P_i \cap A_E} \cup \{w_c\}_{c \in C_i} \cup \{nb\}$,
- $C_i^{TT} = \{c^g\}$,
- $T_i^{TT} = \{\tau_{o_p}\}_{p \in P_i \cap A_E} \cup \{\tau'_p\}_{p \in P_i}$. Such that for each $\tau_p = (l, p, g_{\tau_p}, tc_{\tau_p}, r_{\tau_p}, f_{\tau_p}, l') \in T_i$ we have:

  $\tau_{o_p} = (\perp_{o_p}^l, o_p, True, True, \emptyset, Id, \perp_{o_p}^{l'})$ if $p \in P_i \cap A_E$

  $\tau'_p = (l, p, True, True, r_{\tau_p}, f_{\tau'_p}, l')$,

  where $\perp_{o_p}^{l'}$ is l or $\perp_{o_q}^l$ such that $l \xrightarrow{q}$ and $f_{\tau'_p}$ is as described in Rule 51.
- For places of $L_\perp$, the time progress condition $tpc^{TT}(\perp_{o_p}^l) = tpc(l)$.

Example 1 Figure 8 illustrates transformation of an ATC component into its corresponding ATC TT component. In this example we consider that ports p and q are participating in external interactions.

5.3 Building TTCC Components

As explained before, a TTCC component layer is introduced initially in order to handle intertask interactions and thus model the TT communication system. By considering the need for operational equivalence, and in order to be able to resolve all conflicts of the target model interactions, the TTCC layer handles, on top of intertask interactions, other interactions that are conflicting *directly or indirectly* with these latter. Recall that all interactions of the original model, that are handled in the TTCC layer are called external interactions.

Initially, all components are doing their initial computations and the TTCC layer does not know their state or their enabled communication ports until they send offers. Handling only one external interaction, a TTCC can execute this latter only when all participating tasks' components have sent their offers and are ready to execute the interaction.

Since in the input model we assume that no priority rules can be established between external interactions, a TTCC component doesn't need to connect with tasks participating in interactions other the one it is handling. Since the enabledness of its interaction only depends on offers received from its participating tasks components. When the interaction is conflicting with another external interaction, the TTCC has to communicate, after checking the enabledness of the interaction, with the CRP in order to get the permission or not to execute. We call this communication a reservation mechanism.

To summarize, the behavior of a TTCC component handling an interaction $a = (a, G_a, F_a) \in \gamma$ is made of three steps: (1) it waits for offers from its participating task components, (2) once all offers are received —regardless their order, the TTCC component takes a decision by either executing the interaction upon synchronization (i.e., conjunction of received guards and G_a evaluates to *True*) if a is a non-conflicting interaction or soliciting the CRP component to find out if the conflicting interaction a can be executed and (3) finally it writes on appropriate task components by sending a notification.

Figure 9 shows a representative part of a TTCC automaton, where we can distinguish the three steps. From location *wait*, the TTCC is waiting for respective offers from its participating components. Since these offers can be received in a random order, the TTCC is designed in such a way to allow all possible combination from location *wait*. Once all offers are received, the location *read* is reached. From this location, the TTCC starts the second step in order to execute the interaction depending on whether it is conflicting or not. Once the TTCC executes the interaction, the automaton reaches location *send* from which it executes a transition allowing to notify participating components and reaches back the location *wait*. All transitions of the first step are triggered by receive ports corresponding to respective offers. The transition of the third step is triggered by a send port. Behaviour and ports triggering transitions of the second step are detailed later.

Let a TTCC component handling an external interaction $\alpha = (P_\alpha, G_\alpha, F_\alpha) \in \gamma \cap A_E$. We denote by n the number of components related to TTCC, i.e. the number of participating components of α.

In the case when α is a non-conflicting interaction, the execution of this latter is performed without requesting the CRP component. As shown in Figure 10a, the TTCC executes a transition from location *read* to *send* labelled by a unary port denoted p_α. Its update function executes the update function F_α of the interaction α, and then respective update functions that are received in offers. The transition p_α is guarded by the conjunction of the guard G_α and respective guards and timing constraints received in offers. If the conjunction of these guards evaluates to *True*, the interaction is executed and the TTCC sends a notification to participating components.

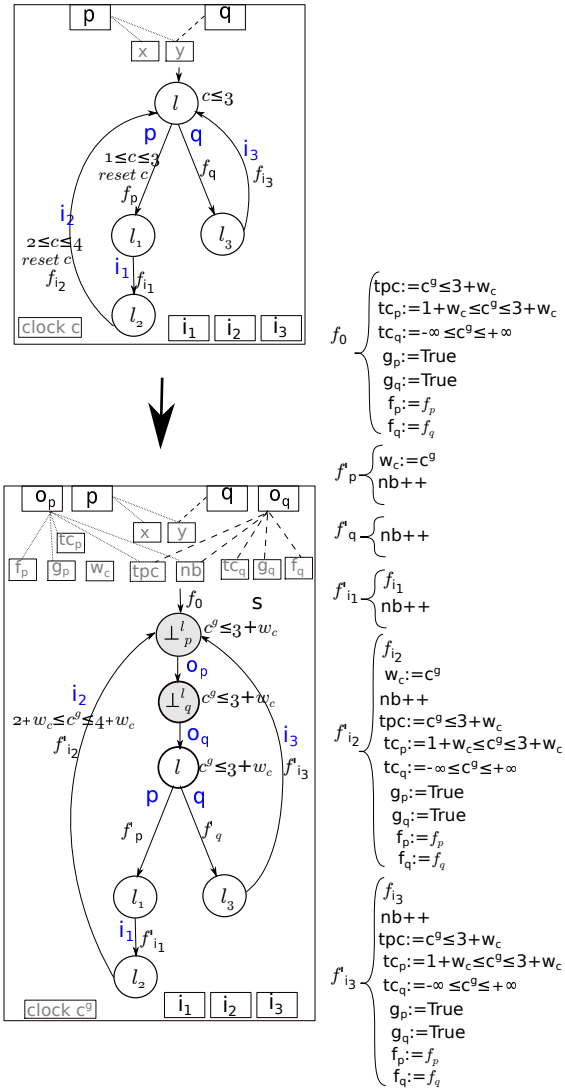


Fig. 8: Example of transformation of an ATC component

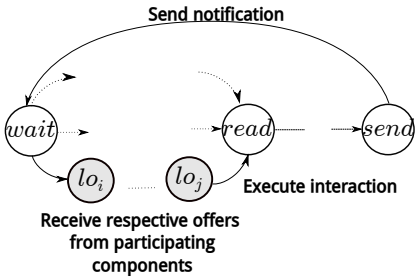
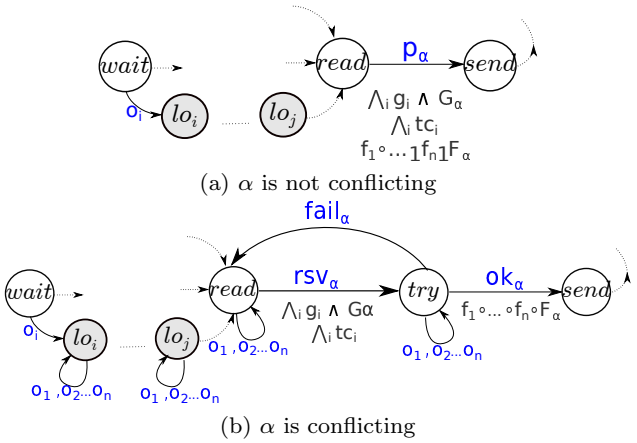


Fig. 9: Skeleton of a TTCC automaton

In the case when α is conflicting with another interaction, the TTCC goes through a reservation mechanism (cf. Figure 10b). If the interaction is enabled, i.e. the conjunction of the guard G_α and respective guards and timing constraints received in offers evaluates to *True*, the TTCC executes transition rsv_α from location *read*. This transition reaches location *try*. By the execution of rsv_α, a reservation request is sent to the CRP component. This reservation contains different values of participation count variables of α participating components. Based on these participation counters, the CRP decides whether to allow or disallow the interaction execution. It notifies the TTCC component either through port ok_α in the case when the reservation succeeds or through port $fail_\alpha$ if the reservation can not be made. While waiting for CRP notification, the TTCC occupies the location *try*. If the port ok_α is enabled, then it executes the transition reaching location *send* from which notification to components are ready to be sent. Note that update function F_α composed with those of received offers is associated with the transition labelled by the ok_α port. If the port $fail_\alpha$ is enabled, the TTCC

reaches back the location *read* in order to proceed again for the reservation.



(a) α is not conflicting

(b) α is conflicting

Fig. 10: Mechanisms for execution of interaction $\alpha = (P_\alpha, G_\alpha, F_\alpha)$

When an ATC component is participating in two conflicting interactions α_1 and α_2, it sends successively offers to each of the corresponding TTCC components $TTCC_{\alpha_1}$ and $TTCC_{\alpha_2}$ and waits from a notification from one of them. After resolving the conflict by requesting the CRP, suppose $TTCC_{\alpha_1}$ will notify the component after successfully executing the interaction α_1, while $TTCC_{\alpha_2}$ reaches back its location *read* in order to proceed to a new reservation attempt. The component is able to continue execution of its next transitions. And it may reach again the location allowing to send again offers to $TTCC_{\alpha_1}$ and $TTCC_{\alpha_2}$. Both TTCC components should be ready to receive the offers. For that, we add loop transitions in TTCC automata labelled by offers receive ports over locations *read* and *try*. Furthermore, such an ATC component may need to resend an offer to a TTCC even before this latter receives other offers from the rest of its participating components. This is resolved by adding loop transitions labelled by offer receive ports over locations that are placed between location *wait* and *read* (cf. Figure 10b). These added loop transitions allow to respect the last point of Definition 7 stating that whenever a send port is activated, all its receive ports are enabled as well.

5.3.1 Formal Transformation rules

In the following, we explicit the transformation rule allowing to instantiate a TTCC components for each external interaction. Each external interaction $\alpha = (P_\alpha, G_\alpha, F_\alpha) \in \gamma \cap A_E$, such that $P_\alpha = \{p_i\}_{i\in[1,n]}$, and $comp(\alpha) = \{B_i\}_{i\in[1,n]}$, is transformed into a TTCC component $TTCC = (L^{TTCC}, P^{TTCC}, X^{TTCC}, C^{TTCC}, T^{TTCC}, tpc^{TTCC})$.

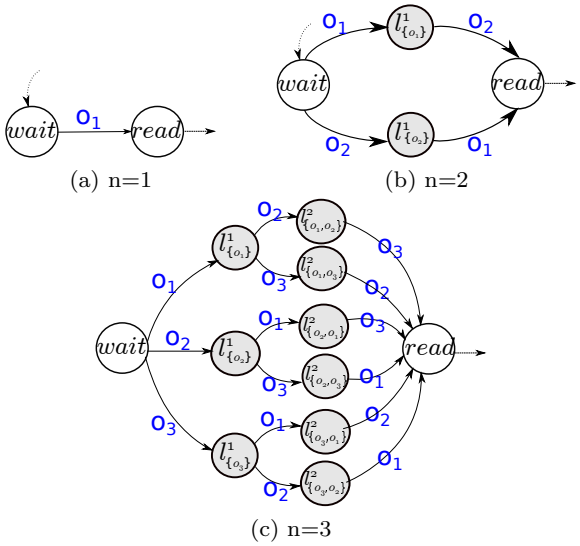
Rule 52 (Ports P^{TTCC} and variables X^{TTCC})

- *For each port $p_i \in P_\alpha$, we include in P^{TTCC} a receive port o_{p_i}. For each port o_{p_i} we associate a local copy of the set of variables X_{p_i} initially exported by port p_i of component B_i. We associate also to o_{p_i} the time progress condition variable tpc_i,the timing constraint variable tc_{p_i}, the Boolean guard variable g_{p_i}, the update function variable f_{p_i} and the participation count variable nb_i.*
- *We include also one send port p_s^α in P^{TTCC}. To the port p_s^α, we associate sets of local variables X_{p_i}, $p_i \in P_\alpha$.*
- *If α is not conflicting, then we include a unary port denoted p_α, which allows to label the transition executing the interaction. Otherwise, we include in P^{TTCC} one send port rsv_α and two receive ports ok_α and $fail_\alpha$. Only port rsv_α has associated variables, which are participation count variables nb_i for all $i \in [1,n]$, i.e. all participation count variables of participating components $\{B_i\}_{i\in[1,n]}$*

Rule 53 (Clock set C^{TTCC}) *As explained before, the TTCC component defines only one clock which is the global clock denoted c^g.*

Rule 54 (Locations set L^{TTCC} and tpc^{TTCC})

- *We include in L^{TTCC} location* wait *marking thee beginning of offer reception, location* read *marking the reception of all offers and the location* send *marking the end of interaction execution. If $n \geq 2$, we include —between location* wait *and* read*—the set of intermediate waiting locations $L_\perp$ allowing reception of offers in any order.*
  Let $\mathcal{O} = \{o_{p_i} \,|\, p_i \in P_\alpha, i \in [1,n]\}$ be the set of all offers received by TTCC. The set $L_\perp$ is constructed as follows; $L_\perp = \{l^k_{O_k} \,|\, k \in [1, n-1], O_k \in \mathcal{P}_{k(\mathcal{O})}\}$, where $\mathcal{P}_k(\mathcal{O})$ is the k-permutation of $\mathcal{O}$, allowing to indicate the ordered subset of offers sent before reaching the location $l^k_{O_k}$. Note that the cardinality of $L_\perp$ is $|L_\perp| = \sum_{k=1}^{n-1} \frac{n!}{(n-k)!}$. Figure 11 shows how intermediate waiting locations (displayed in gray) are constructed for $n = 2$ and $n = 3$. Its shows also the case when $n = 1$, where no intermediate waiting location is needed.
- *If α is conflicting, we introduce in L^{TTCC} the location try allowing the reservation mechanism.*
- *The time progress condition of location* wait *is set to True. The time progress condition of location send is False. In the case of a conflicting TTCC, the time progress condition of its try is True. For location*



(a) n=1 (b) n=2 (c) n=3

Fig. 11: Intermediate waiting locations

read, *the time progress condition is set to the conjunction of time progress conditions received in the offers. That is, after receiving offers from participating components, we require that the TTCC component executes its interaction before different time progress conditions of participating components become False.*

Rule 55 (Transitions set T^{TTCC})

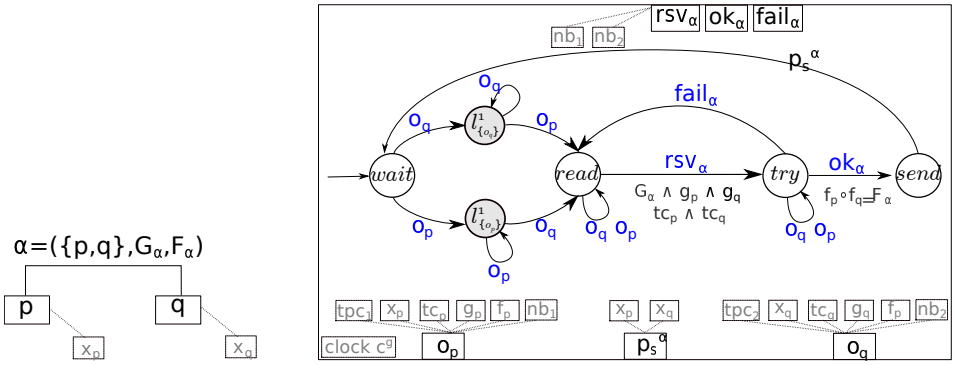
- *In order to receive offers from task components B_i, we include receiving transition, we have three classes of receiving transitions; the n transitions starting from location wait and labelled each by an offer port, transitions between locations $L_\perp$ and transitions reaching the location read. They are respectively as follows:*

$$\begin{aligned}
\tau_{o_{p_i}} =& (wait, o_{p_i}, True, True, \emptyset, Id, l^1_{O_1}),\\
& \forall O_1 \in \mathcal{P}_1(\mathcal{O}) :\ o_{p_i} \in O_1\,,\\
\tau_{o_{p_i}} =& (l^k_{O_k}, o_{p_i}, True, True, \emptyset, Id, l^{k+1}_{O_{k+1}}),\\
& \forall k \in [1, n-2] :\ O_k \subsetneq O_{k+1}, o_{p_i} \in O_{k+1} \setminus O_k\,,\\
\tau_{o_{p_i}} =& (l^{n-1}_{O_{n-1}}, o_{p_i}, True, True, \emptyset, Id, read),\\
& \forall O_{n-1} \in \mathcal{P}_{n-1}(\mathcal{O}) :\ o_{p_i} \notin O_{n-1}\,.
\end{aligned}$$

  These transitions' guards and timing constraints are default to True, their update functions are the identity function and they does not reset clocks.
- *If α is conflicting, the set of transitions includes loop waiting transitions as already explained, for each $l^k_{O_k} \in L_\perp$, we include k loop transitions labelled each by an offer port $o_{p_i} \in O_k$. That is, for each $l^k_{O_k} \in L_\perp$, and for each $o_{p_i} \subsetneq O_k$, we include the transition $\tau^{l^k_{O_k}}_{o_{p_i}} = (l^k_{O_k}, o_{p_i}, True, True, \emptyset, Id, l^k_{O_k})$. we add also loop transitions on locations* read *and* try*, i.e. for each $o_{p_i} \in \mathcal{O}$, we add $\tau^{read}_{o_{p_i}} = (\text{read}, o_{p_i}, True, True, \emptyset, Id, \text{read})$ and $\tau^{try}_{o_{p_i}} = (\text{try}, o_{p_i}, True, True, \emptyset, Id, \text{try})$. These transitions allow components participating in conflicting interactions that have already sent their offer to be able to send it again.*
- *To notify task components after executing the interaction α, we include the transition $\tau_{send} = (send, p^\alpha_s, True, True, Identity, \emptyset, wait)$.*
- *If α is not conflicting, we include the transition $\tau_\alpha = (read, p_\alpha, G^*, TC^*, \emptyset, F^*, write)$, where the port p_α is a unary port, $G^* = G_\alpha \bigwedge(\bigwedge_{i=1}^{n} g_{p_i})$, $TC^* = \bigwedge_{i=1}^{n} tc_{p_i}$, $F^* = f_{p_1} \circ ... \circ f_{p_n} \circ F_\alpha$ such that G_α and F_α are respectively the guard and the update function of the initial interaction α, g_{p_i}, tc_{p_i} and f_{p_i} are respectively the guard, the timing constraint and the update function of offer o_{p_i}.*
- *If α is conflicting, we include transitions allowing the reservation mechanism: $\tau_{rsv} = (read, rsv, G^*, TC^*, \emptyset, Id, try)$,*
  $\tau_{ok} = (try, ok, True, True, \emptyset, F^*, send)$,
  $\tau_{fail} = (try, fail, True, True, \emptyset, Id, read)$, where G^*, TC^* and F^* are as detailed in the previous item.

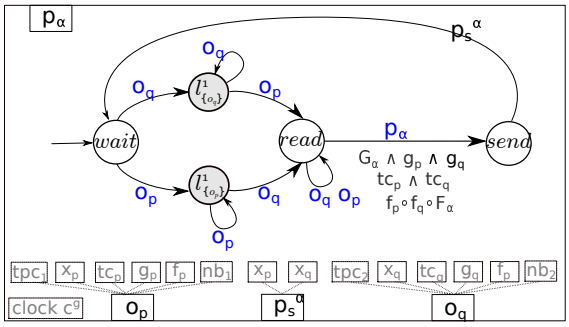
Example 2 In Figure 12b (resp. Figure 12c), we illustrate transformation of a conflicting (resp. non conflicting) external interactions α into its corresponding TTCC component. In these examples we consider that ports p and q of the interaction α are exporting respectively variables x_p and x_q.

The conflict resolution protocol (CRP) that we use in our work is the same CRP used in [10,14,15]. It is, so far, the unique solution to guarantee the resolution of conflicts without requesting the BIP execution engine. It accommodates the algorithm proposed in [5]. It uses message counts to ensure synchronization and reduces the conflict resolution problem to dining or drinking philosophers [8]. Its main role is to check the freshness of requests received for an interaction, that is, to check that no conflicting interactions have been already executed using the same request. In each request, an interaction sends the participation numbers of its components, i.e. number of interactions each ATC component has participated in. This ensures that two conflicting interactions cannot execute with the same request. Mutual exclusion is ensured using participation numbers. To this end, the conflict resolution protocol keeps the last participation number NB_i of each component B_i and compares it with the participation number nb_i provided along with the reservation request from TTCC components. If each participation number from the re-



(a) Original interaction

(b) TTCC component of a conflicting interaction



(c) TTCC component of a non-conflicting interaction

Fig. 12: Example of transformation of an interaction into a TTCC component

quest is greater than the one recorded by the conflict resolution protocol ($nb_i > NB_i$), the interaction is then granted to execute and NB_i is updated to nb_i. Otherwise, the interaction execution is disallowed.

Example 3 Figure 13 presents the places, transitions, variables, guards and update functions involved in handling an interaction α with two participating components B_1 and B_2. Whenever a reservation for executing α arrives, the place r_α is reached. From this place, if the guard of the transition labelled by ok_α is *True*according to the current values of NB_i variable and freshly received nb_i variables, the transition can take place. The transition labelled by $fail_\alpha$ is always possible.

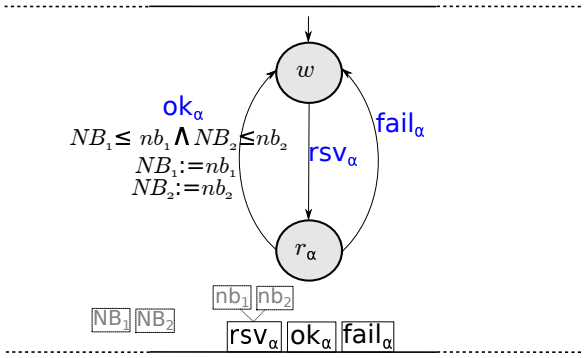


Fig. 13: Fragment of the CRP component

5.4 Building the CRP Component

5.4.1 Formal Transformation rule

In the following, we explicit the rule allowing to instantiate a CRP component.

Rule 56 *Given the model* $B \stackrel{def}{=} \gamma(B_1, ..., B_n)$*, we instantiate the component* $CRP = (L^{CRP}, P^{CRP}, X^{CRP}, C^{CRP}, T^{CRP}, tpc^{CRP})$ *where:*

- L^{CRP} *contains the waiting place* w *where* $tpc(w) = True$*;*
- X^{CRP} *contains the last used offer variable* N_i *for each* $B_i \in comp(\alpha)$ *where* $\alpha \in A_E$*,*
- $C^{CRP} = c^g$*,*
- *For each externally conflicting* $\alpha \in A_E$*,*
  - L^{CRP} *contains the reservation place* r_α *where* $tpc(r_\alpha) = False$*,*
  - P^{CRP} *contains the ports* rsv_α*,* ok_α *and* $fail_\alpha$*,*
  - X^{CRP} *contains the participation numbers* $\{nb_i^\alpha \mid B_i \in comp(\alpha)\}$*. These variables are associated to the port* rsv_α*. Ports* ok_α *and* $fail_\alpha$ *do not have associated variables.*
  - T^{CRP} *contains the following three transitions;* $\tau_{rsv_\alpha} = (w, rsv_\alpha, r_\alpha)$*,* $\tau_{ok_\alpha} = (r_\alpha, ok_\alpha, w)$ *and* $\tau_{fail_\alpha} = (r_\alpha, fail_\alpha, w)$*. The transitions* τ_{rsv_α} *and* τ_{fail_α} *has no guard, no timing constraint and no*

update function. The transition τ_{ok_α} has no timing constraint but is guarded by

$$G_{\tau_{ok_\alpha}} = \bigwedge_{B_i \in comp(\alpha)} nb_i^\alpha > NB_i\,.$$

Its update function sets the variables NB_i of components $B_i \in comp(\alpha)$ to the values of corresponding participation numbers nb_i^α: i.e. for each $B_i \in comp(\alpha)$, it performs $NB_i := nb_i^\alpha$.

5.5 Cross-layer interactions

In this section, we define the interactions between the task components and the TTCC layer and between this latter and the CRP component. Tasks and TTCC components exchange offers and notifications. Communication between TTCC components and the CRP component involves the transmission of messages corresponding to rsv, ok and $fail$ (cf. Rule 57). In the following rule, and for clarity of presentation, we use the notation $B.p$ to denote the port p of the component B.

Rule 57 *Let $B \stackrel{def}{=} \gamma(B_1, ..., B_n)$ be a BIP model, $\mathcal{T}$ be a task mapping. We define the obtained model after transformation as $B^{TT} = \gamma^{TT}(B_1^{TT}, ..., B_n^{TT}, TTCC_1, ..., TTCC_m, CRP)$. The send/receive interactions of γ^{TT} are defined as follows:*

- *For each task component $B_{T_j}^{TT}$ such that $T_j \in \mathcal{T}$, for each port $B_{T_j}^{TT}.o_p$ and each $TTCC_\alpha$ such that $p \in \alpha$, we include in γ^{TT} the offer interaction based on ports $(B_{T_j}^{TT}.o_p, TTCC_\alpha.o_p)$. Its guard is set to True. And its update function copies variables associated with $B_{T_j}^{TT}.o_p$ to those of the receive port $TTCC_\alpha.o_p$.*
- *For each $TTCC_\alpha$, and all $\{B_{T_j}^{TT}\}_{j\in J}$, such that for all $j \in J$, $T_j \cap comp(\alpha) \neq \emptyset$, we include the notification interaction based on ports $(TTCC_\alpha.p_s^\alpha, \{B_{T_j}^{TT}.p_j\}_{j\in J})$, where for all $j \in J$, $p_j \in \alpha$. Its guard is set to True. And its update function copies variables associated with $TTCC_\alpha.p_s^\alpha$ to those of the receive ports $B_{T_j}^{TT}.p_j$.*
- *For each interaction $\alpha \in \gamma$ that is not conflicting, we include the unary interaction having as unique port $(TTCC_\alpha.p_\alpha)$, where $TTCC_\alpha$ is the TTCC component handling the interaction α. Its guard is set to True. And its update function is the identity function.*
- *For each interaction $\alpha \in \gamma$ that is conflicting, we include a triplet of interactions having respectively the following sets of ports: $(TTCC_\alpha.rsv_\alpha, CRP.rsv_\alpha)$, $(CRP.ok_\alpha, TTCC_\alpha.ok_\alpha)$ and $(CRP.fail_\alpha, TTCC_\alpha.fail_\alpha)$. All their guards are set to True. The update function of the former interaction copies variables of ports $TTCC_\alpha.rsv_\alpha$ to port $CRP.rsv_\alpha$. Since ports $CRP.ok_\alpha$ and $CRP.fail_\alpha$ do not have any associated variables, the update function of the last two interactions is the identity function.*

6 Transformation Correctness

In this section, we show that the described transformation is correct, that is the obtained TT-BIP model is observationally equivalent to the original BIP model. Before proving the observational equivalence, we show that the final model is a valid TT-BIP model.

6.1 Validity of the Obtained Model

Proposition 1 *Given a BIP model $B = \gamma(B_1, \ldots, B_n)$ and a task mapping $T = \{T_1, \ldots, T_k\}$, the model $B^{TT} = \gamma^{TT}(B_1^{TT}, \ldots, B_n^{TT}, TTCC_1, \ldots, TTCC_m, CRP)$ obtained by the transformation in Section 5 meets the properties of Definition 7.*

A proof of Proposition 1 is provided in Appendix A. This proof ensures that any component ready to perform a transition labelled by a send port will not be blocked by waiting for the corresponding receive ports.

6.2 Observational Equivalence Between B and B^{TT}

We denote $B = \gamma(B_1, \ldots, B_n)$ the initial model and $B^{TT} = \gamma^{TT}(B_1^{TT}, \ldots, B_n^{TT}, TTCC_1, \ldots, TTCC_m, CRP)$ the resulting model of the first step of the transformation.

In order to prove the correctness of the transformation from B to B^{TT}, we have to show that their corresponding semantic LTSs are observationally equivalent. We denote by $G(B)$ and $G(B^{TT})$ successively the LTSs of B and B^{TT} (see Definition 6).

We define observational equivalence between transition systems based on the classical notion of weak bisimilarity [13], where some transitions are considered unobservable.

We will use the following notation. Consider a binary relation $R \subseteq X \times Y$. For $x \in X$, we denote $R(x) \stackrel{def}{=} \{y \in Y \mid (x, y) \in R\}$.

Definition 10 (LTS relations) Let $A = (Q_A, P_A, \underset{A}{\longrightarrow})$ and $B = (Q_B, P_B, \underset{B}{\longrightarrow})$ be two LTS. Given a relation $\beta \subseteq P_A \times P_B$, we write $q \xrightarrow[A]{\beta} q'$, for $q \in Q_A$, iff there exists $a \in P_A$, such that $q \xrightarrow[A]{a} q'$ and a is *not* related by β to any label in P_B, i.e. $\beta(a) = \emptyset$. The notation $q \xrightarrow[B]{\beta} q'$, for $q \in Q_B$, is defined symmetrically.

A *weak simulation* over A and B, is a pair of relations $R \subseteq Q_A \times Q_B$ and $\beta \subseteq P_A \times P_B$, such that:

$$\forall (q,r) \in R,\, \forall a \in P_A,\, \Big(\beta(a) \neq \emptyset \wedge q \xrightarrow[A]{a} q' \implies \exists (a,b) \in \beta : \exists (q',r') \in R : r \xrightarrow[B]{\beta^* b \beta^*} r'\Big)$$

and

$$\forall (q,r) \in R,\, \Big(q \xrightarrow[A]{\beta} q' \implies \exists (q',r') \in R : r \xrightarrow[B]{\beta^*} r'\Big),$$

where β^* denotes zero or more successive β transitions (i.e. transitions whose label is not related by the relation β).

A *weak bisimulation* over A and B is a pair of relations $R \subseteq Q_A \times Q_B$ and $\beta \subseteq P_A \times P_B$, such that both (R,β) and (R^{-1},β^{-1}) are weak simulations, where $R^{-1} \subseteq Q_B \times Q_A$ and $\beta^{-1} \subseteq P_B \times P_A$ are the inverse relations of R and β, respectively.

We say that A and B are *weakly bisimilar w.r.t.* $\beta \subseteq P_A \times P_B$, denoted $A \sim_\beta B$, if there exists $R \subseteq Q_A \times Q_B$ total on both Q_A and Q_B, such that (R,β) is a weak bisimulation.

First, we have to establish a correspondence between labels of $G(B)$ (ranging over the set $\gamma \cup \mathbb{R}_+$) and those of $G(B^{TT})$ (ranging over the set $\gamma^{TT} \cup \mathbb{R}_+$). Therefore, we define the relation β as follows:

$$\beta = \big\{(\alpha,\alpha) \,|\, \alpha \in \gamma \cap A_I\big\} \cup \big\{(\alpha, p_s^\alpha) \,|\, \alpha \in \gamma \cap A_E\big\}, \tag{8}$$

where p_s^α is the send port of the TTCC component allowing to send notifications to its related components.

Note that by this relation, we can say that each transition $\alpha \in \gamma$, is represented in γ^{TT} either by the transition α itself if it is internal, or by p_s^α if it is external. Transitions of B that are not related by the relation β are only delay transitions. And transitions of B_{TT} that are not related by the relation β are offer, reserve, fail, ok and p_α transitions.

We may use later in this proof the notations $fail_\alpha$, ok_α and rsv_α to denote, respectively, the fail, ok and reservation interactions between the CRP and the TTCC component handling interaction α in B^{TT} model.

Theorem 1 *The LTSs $G(B)$ and $G(B^{TT})$ are weakly bisimilar w.r.t. β, i.e. $G(B) \sim_\beta G(B^{TT})$.*

A proof of this theorem is provided in Appendix B.

7 Implemantation and use case

The transformation rules have been implemented into BIP toolset as an eclipse plugin called BIP2TT-BIP tool. The case study used to validate the implemented tool is the Flight Simulator (FS) application [6] dedicated to the navigation of DIY radio controlled planes. The original application is written in Modelica [9].

This application provides a simulation of the physics of a plane and an automatic pilot who tries to reach given way-points on a map. The simulation of the Modelica model gives a display of the road followed by the plane (specifically the trajectories of left and right wingtips).

The Modelica model consists of a set of six communicating sub-models (cf. Figure 15): autopilot, fly-by-wire, route planner, servo (i.e. the actuator), simulator and sensor. The autopilot models the pilot commands in function of the flight state. It has four main functionalities: flight state reception from sensor component, execution of the route planner, execution of fly-by-wire and sending command to servo component. The route component sends information to fly-by-wire after computing distance to current waypoint and changing route towards next waypoint if necessary. It operates in low frequency: every 15 seconds. The fly-by-wire component allows course correction by setting roll attitude and ailerons and elevator. It operates in high frequency: every 5 seconds. The servo refers to the autopilot's actuation on plane's flight control surfaces. Servo component receives command from autopilot component and transfers it to simulator component. Some filtering (e.g. low-pass, delay) could be added to mimic realistic actuators. The Flight simulator simulates flight dynamics computation of plane and wing tips position based on received commands (i.e. new values of roll, pitch and throttle). The sensor refers to the autopilot's perception of real world data. Sensor component receives data about flight state from simulator component and resends them to the autopilot. The sensor can add some noise (e.g. delay, etc.) to mimic realistic data acquisitions. But in our example, it stands for copying the state computed by simulator component.

These submodels are communicating through Modelica connectors. The software architecture of the original Modelica model is shown in Figure 14.

We have first modelled the FS application in BIP language. This latter —coupled with different task mapping strategies—is the input of transformation tools. We also simulate the initial BIP model, the TT-BIP model (the output of the BIP2TT-BIP tool) in order to compare their respective behavior.

Each sub-model of the modelica model is modelled as a BIP component, communication between different components is modeled using BIP connectors. In Figure 15, the overall architecture of the BIP model is displayed. The bihavior of each component is modeled with a timed automata. We apply the transformation of the BIP2TT-BIP tool in order to derive the TT-

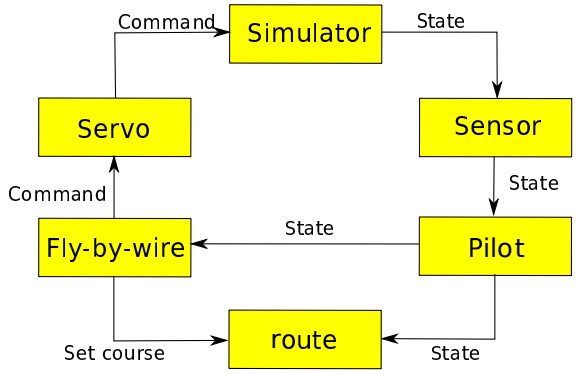


Fig. 14: Software Architecture of the Modelica Model of the Flightsim Application

BIP model following different task mapping strategies. In this paper we consider the task mapping strategy $TM1 : T_1 = \{FLY\}, T_2 = \{ROUTE\}, T_3 = \{PILOT\}, T_4 = \{SERVO\}, T_5 = \{SIMULATOR\}$ and $T_6 = \{SENSOR\}$.

Figure 16, shows the obtained model for the task mapping $TM1$. For clarity reason, behaviours of TTCC and CRP components are not displayed. Nonetheless, since all TTCC components are connecting exactly two tasks, their automata are strictly similar to those of Figure 12b and Figure 12c.

In order to be able to compare the functionality of the original BIP model with the obtained TT-BIP model, we use BIP simulator that generates C++ code from the original and the TT-BIP models. Simulation of two generated C++ codes allowed us to visualize and compare the output signals. A band shows the trajectories of left and right wingtips and illustrates the roll movement that precedes the change in course at each waypoint, while the plane progressively reaches its desired altitude. Figure 17 presents the simulation results of the initial and the derived models, for the waypoints (300,0,300), (300,300,300), (0,300,300) and (0,0,300). The inspection reveals that the output of the transformed model is strictly similar to that of the original model.

8 Conclusion

In this paper, we have presented a model to model transformational method allowing to explicit TT communication settings in the obtained model. The obtained model is structured following the TT-BIP architecture. It consists of tasks layer, communication layer and the conflict resolution layer. The first layer is obtained after transforming components participating in external interactions depending on a user-defined task mapping. Each TTCC component of the second layer is dedicated to handle one external interaction and communicate with tasks of the layer underneath in two steps; it receives offers and sends notification after executing the interaction. The third layer is responsible of resolving conflicts between different interactions handled by the second layer.

The obtained model is based on one global clock, implements multiparty interactions through dedicated communication media (i.e. TTCC components) and ensures communication between different layers by using message passing interactions (i.e. Send/receive interactions). Even though the obtained model satisfies the TT settings described in the opening of Section 3, it is yet still far from being intuitively translatable to the programming language of a target platform which is based on the TT execution model.

In an ongoing work, we present a method for generating TT implementation from the obtained TT-BIP model.

References

1. Abdellatif, T.: Rigourous implementation of real-time systems. Ph.D. thesis, UJF (2012)
2. Abdellatif, T., Combaz, J., Sifakis, J.: Model-based implementation of real-time applications pp. 229–238 (2010)
3. Alur, R., Dill, D.L.: A theory of timed automata. Theoretical computer science **126**(2), 183–235 (1994)
4. Aussagues, C., Chabrol, D., David, V., Roux, D., Willey, N., Tournadre, A., Graniou, M.: Pharos, a multicore os ready for safety-related automotive systems: results and future prospects. Proc. of The Embedded Real-Time Software and Systems (ERTS2) (2010)
5. Bagrodia, R.: Process synchronization: Design and performance evaluation of distributed algorithms. Software Engineering, IEEE Transactions on **15**(9), 1053–1065 (1989)
6. Ben Hedia, B., Hamelin, E.: Projet openprod rapport r4.28 : Model to embedded real-time transformation. Tech. rep. (2012)
7. Boulanger, J.L., Fornari, F.X., Camus, J.L., Dion, B.: SCADE: Language and Applications. Wiley-IEEE Press (2015)
8. Chandy, K.M., Misra, J.: The drinking philosophers problem. ACM Transactions on Programming Languages and Systems (TOPLAS) **6**(4), 632–646 (1984)
9. Elmqvist, H., Mattsson, S.E.: An introduction to the physical modeling language modelica. In: Proceedings of the 9th European Simulation Symposium, ESS, vol. 97, pp. 19–23. Citeseer (1997)
10. Jaber, M.: Centralized and distributed implementations of correct-by-construction component-based systems by using source-to-source transformations in bip. Theses, Université Joseph-Fourier - Grenoble I (2010). URL https://tel.archives-ouvertes.fr/tel-00531082
11. Kaiser, R., Wagner, S.: Evolution of the pikeos microkernel. In: Proceedings of the 1st International Workshop on Microkernels for Embedded Systems, pp. 50–57 (2007)
12. Kopetz, H.: The time-triggered approach to real-time system design. Predictably Dependable Computing Systems. Springer (1995)
13. Milner, R.: Communication and Concurrency. Prentice Hall International (UK) Ltd., Hertfordshire, UK, UK (1995)

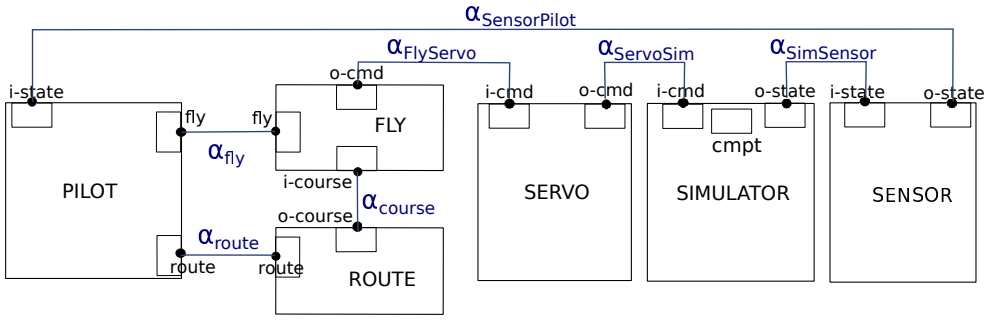


Fig. 15: Initial Flightsim BIP model

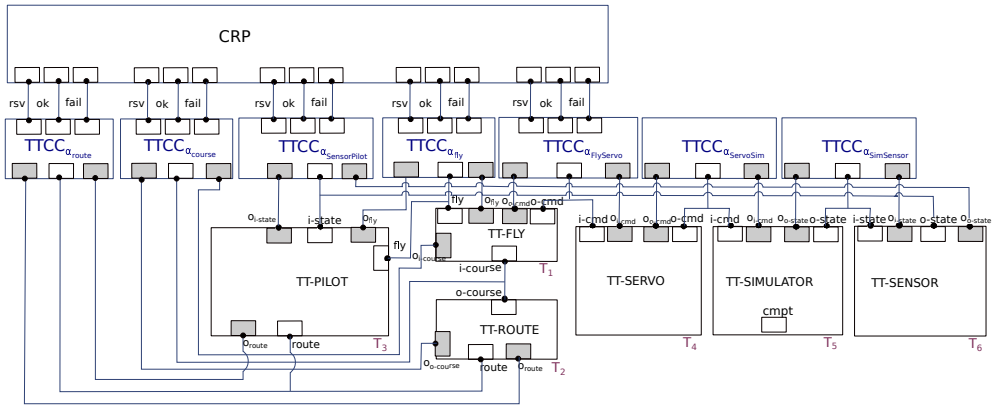


Fig. 16: FS TT-BIP Model for the Task mapping TM1

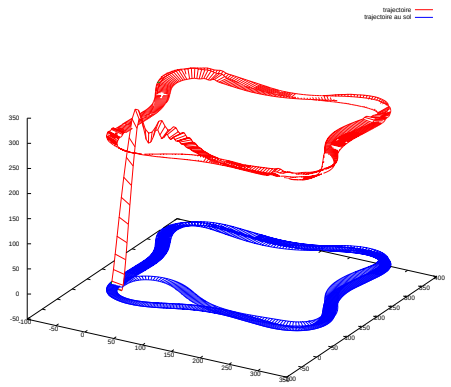


Fig. 17: Trajectories of left and right wingtips

14. Quilbeuf, J.: Distributed implementations of component-based systems with prioritized multiparty interactions. application to the bip framework. Ph.D. thesis, Université de Grenoble (2013)

15. Triki, A.: Distributed implementations of timed component-based systems. Ph.D. thesis, Grenoble Alpes (2015)

Appendices

A Proof of Proposition 1

Proof **Points 1-3 of Definition 7**
The first three criteria of Definition 7 are syntactic, namely only allowed interactions are either classic multiparty interactions or send/receive interactions or unary interactions and each send port participates in exactly one Send/Receive interaction. These criteria are met by the previous definition.

Point 4 of Definition 7
The fourth point of Definition 7, enumerates all conflict cases of a TT-BIP model. The first case states that an internal port can only be conflicting with a similar port. By construction of the transformation, internal ports are instantiated only in task components (cf. Rule 51). If an internal transition is originally conflicting with a similar transition then this conflict is preserved, since these transitions remain intact after transformation. If in the original model, an internal transition is conflicting with an external transition then this port will be replaced by a send and receive ports. Therefore, the original conflict is no more existing in TT-BIP.

The second case involves receive ports. In task components, by construction of the transformation (cf. Rule 51), a receive port can be only conflicting with receive port. In TTCC component, receive transitions are offer transitions or *ok/fail* transitions. *Ok* transitions and *fail* transitions have the same source location. Similarly, offer transitions can be also enabled from the same location (in the case of conflicting TTCC component). They also can be conflicting with a send transition labelled by an rsv_α port (cf. Rule 54 and Rule 55).

In CRP component, receive transitions are *rsv* transitions which are enabled from the initial location only simultaneously with other rsv transitions. Therefore, in all components, a receive transition can be enabled simultaneously either with a receive port or with a send port or both.

The third case involves send ports. In task components send ports are *offer* ports and by construction of the transformation (cf. Rule 51) only one send port is enabled from one location. In TTCC components, send ports are either p_s^α ports (sending notifications to task components) or rsv_α ports. The former has no conflicting port (i.e. no other port is enabled from its source location) while the latter is enabled from the same location as receive ports (offer ports) (cf. Rule 54 and Rule 55). In CRP component, send ports are *ok* or *fail* ports. Note that these ports are enabled from the same location. Therefore we deduce that a send port can have the same source location as a receive or other send ports.

Point 5 of Definition 7
The fifth point of Definition 7 states that the update function of a transition labelled by a send port does not involve variables exported by this port. In task components, send ports are *offer* ports and they trigger transitions whose update functions are the identity function (cf. Rule 51). In TTCC components, the send port is either a p_s^α or a rsv_α port. In both cases, it labels a transition with an identity update function (cf. Rule 54 and Rule 55). In the CRP component, send port can be either an *ok* or *fail* port. In the first case, the port labels a transition whose update function applies on NB_i variables which are not exported. In the second case, the port labels a transition with an identity update function.

Point 6 of Definition 7
The second-last point in Definition 7 states that a transition labelled by a receive port always has a timing constraint and guards that are default to *True*. In the layer of task components, receive ports label only notification transitions which, by construction, are associated with a timing constraint and guard equal to *True*(cf. Rule 51). In the TTCC layer, receive ports label either *offer* transitions or *ok/fail* transitions. These latter are also associated with a timing constraint and guard always default to *True*(cf. Rule 54 and Rule 55). In the third layer (i.e. the CRP component), receive ports label rsv transitions, which are also associated with timing constraint and guard always equal to *True*.

Point 7 of Definition 7
The last criterion of Definition 7 states that whenever a send port is enabled, the associated receive ports will unconditionally become enabled within a finite number of transitions in the receiver component. Intuitively, this holds since communications between tasks and
TTCC components, and between TTCC components and CRP component follow a request/acknowledgement pattern. Whenever a component sends a request (via a send port) it enables the receive port to receive acknowledgement. In the following, we detail different configuration cases:

- Communications between a task component B_i^{TT} and a $TTCC_j$ component, for all interactions α involving a component B_i. We denote by $l_{B_i^{TT}}$ the enabled location of B_i^{TT} and by l_{TTCC_j} the active place of $TTCC_j$. We distinguish the following cases:

  Case 1: $l_{B_i^{TT}} = \perp_p^l$ where p is exported by B_i and $l_{TTCC_j} \in \{wait\} \cup L_\perp$.
  In this configuration, the only enabled send port involved in a send/receive interaction is the offer port o_p of B_i^{TT}. Note that the initial state allowing a send/receive interaction between tasks and TTCC components falls in that case. By definition of the configuration, all associated receive ports are also enabled (the $TTCC_j$ component can only execute transitions labelled by receive ports).

  Case 2: $l_{B_i^{TT}} = l$ where l is a place of B_i and $l_{TTCC_j} = \{read\}$.
  This configuration is reached from the first one by executing offer transitions. From this configuration, no send/receive interaction with the task components can be enabled (i.e. no send port is enabled). To send offers, the task component should be in a $\perp_p^l$ location which is not the case.

  Case 3: $l_{B_i^{TT}} = l$ where l is a place of B_i and $l_{TTCC_j} = \{send\}$.
  In this case, the component B_i^{TT} is still in a place l that is not a busy location, and the $TTCC_j$ component is in the *send* place. From that configuration, the enabled send port that is involved in a send/receive interaction with B_i^{TT} is the port p_s^α of the TTCC component. By definition of the configuration, the receive port associated to this send port is the one activated from place l of component B_i^{TT}. Thus, the property holds in that configuration as well. Note that after executing the send/receive interaction with the component B_i^{TT}, the first configuration is reached back.
- Communications between a conflicting $TTCC_j^C$ component with the CRP component, for all conflicting interaction α involving a component B_i. We denote by $l_{TTCC_j^C}$ the enabled location of $TTCC_j^C$ and by l_{CRP} the active set of marked places of CRP. We distinguish the following cases:

  Case 1: $l_{TTCC_j^C} = read$ and $l_{CRP} \ni \{w_\alpha\}$.
  In this case, the unique enabled send port is the port rsv_α of the component $TTCC_j^C$. And by definition of the configuration, the associated receive port of this send port is enabled, i.e. the port rsv_α of component CRP is enabled from place w_α. Thus, the property holds in that configuration as well.

  Case 2: $l_{TTCC_j^C} = try$ and $l_{CRP} \ni \{r_\alpha\}$.
  This case is reached by executing the reservation interaction from the previous configuration. In this case, two send ports are active, ok_α and $fail_\alpha$ of the component CRP. From the enabled location of $TTCC_j^C$ component, the corresponding receive ports associated to these two send ports are enabled as well. Thus, the property holds by-construction in that configuration as well.

B Proof of Theorem 1

Proof Let $G(B) = (Q_B, P, \xrightarrow[B]{})$ and $G(B^{TT}) = (Q_{B_{TT}}, P_{B_{TT}}, \xrightarrow[B^{TT}]{})$. Recall (Definition 4) that state spaces Q_B and $Q_{B_{TT}}$ have each three components: control location, clock and variable valuations. For a given state q, we will denote $v_c(q)$ (resp. $v_x(q)$) its clock (resp. variable) valuation component. Similarly, we denote $l(q)$ the location of a state q.

Below, we will use variables q_B, r_B, ranging over Q_B, and $q_{B_{TT}}, r_{B_{TT}}$, ranging over $Q_{B_{TT}}$ and denote their respective components as follows:

$$q_B = \left(l, v_x(q_B), v_c(q_B)\right),$$
$$r_B = \left(l', v_x(r_B), v_c(r_B)\right),$$
$$q_{B_{TT}} = \left(l_{TT}, v_x(q_{B_{TT}}), v_c(q_{B_{TT}})\right),$$
$$r_{B_{TT}} = \left(l'_{TT}, v_x(r_{B_{TT}}), v_c(r_{B_{TT}})\right).$$

For clarity reasons, for each state $q_{B_{TT}}$, we detail the control location l_{TT} by using the triplet $(l^B_{TT}, l^{TTCC}_{TT}, l^{CRP}_{TT})$ where l^B_{TT} denotes the tuple of active locations of the tasks layer components, l^{TTCC}_{TT} contains the tuple of active locations of all TTCC components of the TTCC layer, and l^{CRP}_{TT} contains enabled locations of the CRP. We recall also that a place l of a model $B = \gamma(B_1, \ldots, B_n)$ is written $l = (l_1, .., l_n)$. The place l^B_{TT} of the tasks components layer of the model B^{TT} is written $l^B_{TT} = (l^{TT}_1, .., l^{TT}_n)$. The place l^{TTCC}_{TT} of the TTCC components layer is written as follows $l^{TTCC}_{TT} = (l^{TTCC}_1, \ldots, l^{TTCC}_m)$ while the place l^{CRP}_{TT} of the CRP component is written as $l^{TTCC}_{TT} \in \{w_\alpha, r_\alpha\}$.

We define the relation $R \subseteq Q_B \times Q_{B_{TT}}$ as follows:

$$R = \left\{ (q_B, q_{B_{TT}}) \;\middle|\; \begin{array}{c} l^B_{TT} \in \{l_i, \perp^{l_i}_{p_i}\}^n, \text{ where } l_i \xrightarrow[B_i]{p_i}, \\ v_c(q_B) = v_c(q_{B_{TT}}), \\ v_x(q_B) = v^*_x(q_{B_{TT}}) \end{array} \right\} \quad (9)$$

where v^*_x is the restriction of v_x to the variables X of the original model B. That is the valuation function v^*_x is defined only over variables which are common between B and B_{TT}. We recall that the notation $l_i \xrightarrow[B_i]{p_i}$ means that port p_i is enabled from place l_i of the component B_i.

Note that in the definition (9) of the relation R, there is no restriction to the location of TTCC and CRP components. This means that we consider all states of these components in the defined equivalence class. That is q_B is equivalent with $q_{B_{TT}}$ whose location is a combination of any location of TTCC and CRP components with the locations l_i or $\perp^{l_i}_{p_i}$ of components B. That is $\forall j \in [1, m], l^{TTCC}_j \in \{wait, l_{o_p}, .., read, try, send\}$ and $l^{CRP}_{TT} \in \{w_\alpha, r\alpha\}$.

Thus,the following four assertions prove that (R, β) is a weak bisimulation:

(i) $\forall (q_B, q_{B_{TT}}) \in R$,

$$q_B \xrightarrow[B]{\beta} r_B \implies \exists (r_B, r_{B_{TT}}) \in R : q_{B_{TT}} \xrightarrow[B_{TT}]{\beta^*} r_{B_{TT}},$$

(ii) $\forall (q_B, q_{B_{TT}}) \in R$,

$$q_{B_{TT}} \xrightarrow[B_{TT}]{\beta} r_{B_{TT}} \implies \exists (r_B, r_{B_{TT}}) \in R : q_B \xrightarrow[B]{\beta^*} r_B,$$

(iii) $\forall (q_B, q_{B_{TT}}) \in R, \forall \alpha \in \gamma$,

$$\beta(\alpha) \neq \emptyset \wedge q_B \xrightarrow[B]{\alpha} r_B \implies \exists (\alpha, \alpha') \in \beta : \exists (r_B, r_{B_{TT}}) \in R : q_{B_{TT}} \xrightarrow[B_{TT}]{\beta^* \alpha' \beta^*} r_{B_{TT}},$$

(iv) $\forall (q_B, q_{B_{TT}}) \in R, \forall k \in K$,

$$\beta^{-1}(k) \neq \emptyset \wedge q_{B_{TT}} \xrightarrow[B_{TT}]{k} r_{B_{TT}} \implies \exists (p, k) \in \beta : \exists (r_B, r_{B_{TT}}) \in R : q_B \xrightarrow[B]{p} r_B.$$

Hereafter, we detail proofs of each of these four points:

(i) In definition (8) of the relation β, only interactions of γ are related to interactions of γ^{TT}. That is for each $\alpha \in \gamma$, $\beta(\alpha) \neq \emptyset$. Therefore if $q_B \xrightarrow[B]{\beta} r_B$, then this transition corresponds to a transition that is not related by the relation β. Therefore, by definition (8) of the relation β, the corresponding transition is not an interaction of γ. It is then a transition labelled by a real number representing a delay transition.
By Definition 6, there is a tpc constraint on location l in B, $tpc(l) = (c^g \leq v)$. That is the tpc constraint of each location l_i of each component B_i of the model B (such that $l = (l_1, .., l_n)$) must satisfy this same condition. Therefore, we have:

$$\begin{array}{l} q_B = \big(l, v_x(q_B), v_c(q_B)\big), \\ r_B = \big(l, v_x(r_B), v_c(r_B)\big), \\ v_x(r_B) = v_x(q_B), \\ v_c(r_B) = v_c(q_B) + \delta, v_c(q_B) + \delta \leq v. \end{array} \quad (10)$$

Note that, depending on the nature of interactions enabled from r_B, two cases should be considered. In the first case, only an internal interaction $\alpha_I \in A_I$ can be enabled from state r_B once β executed. In the second case, only external interactions $\alpha_E \in A_E$ are enabled from r_B. By construction of the definition (9) of R, we have $q_B = \big(l, v_x(q_B), v_c(q_B)\big)$, such that

$$v_c(q_B) = v_c(q_{B_{TT}}) \quad \text{and} \quad v_x(q_B) = v^*_x(q_{B_{TT}}). \quad (11)$$

By construction of the transformation (Rule 54, Rule 54 and Rule 51) the same tpc constraint is mapped in the first case to the place l_{TT} where $l_{TT} = l$. In the second case, the same tpc constraint is mapped to the places l_i and $\perp^{l_i}_{p_i}$ where $p_i \in \alpha_E$ as well as to the place $read$ of the corresponding TTCC (handling the interaction α_E). Thus, after executing the β transition corresponding to the mapped tpc in the B_{TT} model, components do not change their places. And there exist a transition $q_{B_{TT}} \xrightarrow[B_{TT}]{\delta} r_{B_{TT}}$ in B_{TT} where $r_{B_{TT}} = (l'_{TT}, v_x(r_B), v_c(r_B))$ such that:

$$l'^B_{TT} = l, \quad v_c(q_B) = v_c(r_B) + \delta \quad \text{and} \quad v_x(q_B) = v_x(r_B). \quad (12)$$

Combining (10), (11) and (12), we obtain that $v_c(r_{B_{TT}}) = v_c(r_B)$ and $v^*_x(r_{B_{TT}}) = v_x(r_B)$. And we deduce that by definition (9) of the relation R, we have $(r_B, r_{B_{TT}}) \in R$.

(ii) If $(q_B, q_{B_{TT}}) \in R$, $q_{B_{TT}} \xrightarrow[B_{TT}]{\beta} r_{B_{TT}}$, then this transition is not related to any transition in γ by the relation β. Therefore and by definition (8) of the relation β, the transition β is either labelled by a real number representing a delay transition or by a send/receive interaction other than the notification transition or a p_α transition. That is, β corresponds either to a rsv_α, $fail_\alpha$, offer, ok_α, p_α interaction or to a delay step.

Case 1: $\beta \in \{rsv_\alpha, fail_\alpha\}$.

By Definition 6, there is a transition $l_{TT} \xrightarrow{\beta \in \{rsv_\alpha, fail_\alpha\}} l'_{TT}$ in B_{TT}, such that:

$$\begin{array}{l} q_{B_{TT}} = \big(l_{TT}(q_{B_{TT}}), v_x(q_{B_{TT}}), v_c(q_{B_{TT}})\big), \\ r_{B_{TT}} = \big(l'_{TT}(r_{B_{TT}}), v_x(r_{B_{TT}}), v_c(r_{B_{TT}})\big), \\ v_x(r_{B_{TT}}) = v_x(q_{B_{TT}}), \quad \text{and} \quad v_c(r_{B_{TT}}) = v_c(q_{B_{TT}}). \end{array} \quad (13)$$

Note that both rsv_α and $fail_\alpha$ define no update function nor a guard or timing constraints (see Rule 57).
By definition of the transformation rules (Rule 54, Rule 55 and Rule 56), in the case of a rsv_α (resp. $fail_\alpha$) interaction, the corresponding TTCC component is in a $read$

(resp. try) place and the CRP component is in w_α (resp. r_α) place. After executing this rsv_α (resp. $fail_\alpha$) transition, the TTCC component reaches place try (resp. $read$) and the place r_α (resp. w_α) is activated in the CRP. Note that, in both cases, places of other components remain intact. That is, the reached place ${l'}^B_{TT} = l^B_{TT} = l$. Thus, we have :

$$ {l'}^B_{TT} = l = (l_1, .., l_n)\,, \tag{14} $$

By construction (9) of R, we have $q_B = \big(l, v_x(q_B), v_c(q_B)\big)$, such that

$$ v_c(q_B) = v_c(q_{B_{TT}}) \quad \text{and} \quad v_x(q_B) = v^*_x(q_{B_{TT}})\,. \tag{15} $$

Combining (13) and (15) we obtain that $v_c(r_{B_{TT}}) = v_c(q_B)$ and $v^*_x(r_{B_{TT}}) = v_x(q_B)$. Combining this to (14), we deduce that by definition (9) of the relation R, we have $(q_B, r_{B_{TT}}) \in R$.

Case 2: β is an offer interaction.

By Definition 6, there is a transition $l_{TT} \xrightarrow{\beta} l'_{TT}$ in B_{TT}, where β allows sending an offer from port p_i of component B_i to the corresponding TTCC component, such that:

$$ \begin{aligned} q_{B_{TT}} &= \big(l_{TT}, v_x(q_{B_{TT}}), v_c(q_{B_{TT}})\big)\,,\\ r_{B_{TT}} &= \big(l'_{TT}, v_x(r_{B_{TT}}), v_c(r_{B_{TT}})\big)\,,\\ v_x(r_{B_{TT}}) = v_x(q_{B_{TT}}), &\quad \text{and} \quad v_c(r_{B_{TT}}) = v_c(q_{B_{TT}})\,. \end{aligned} \tag{16} $$

Note that the offer transition defines no update function nor a guard or timing constraint (see Rule 57).
By definition of the transformation rules (Rule 54, Rule 55 and Rule 56), after executing this β transition, the TTCC component reaches a place l_{o_i} and the component B_i reaches a place $\perp^{l_i}_{p'_i}$ if another offer is likely to be sent, otherwise it reaches the place l_i. Note that this β transition does not change the location of the CRP component. Thus, we have :

$$ {l'}^B_{TT} \in \{l_i, \perp^{l_i}_{p_i}\}^n\,. \tag{17} $$

By construction (9) of R, we have $q_B = \big(l, v_x(q_B), v_c(q_B)\big)$, such that

$$ v_c(q_B) = v_c(q_{B_{TT}}) \quad \text{and} \quad v_x(q_B) = v^*_x(q_{B_{TT}})\,. \tag{18} $$

Combining (16) and (18) we obtain that $v_c(r_{B_{TT}}) = v_c(q_B)$ and $v^*_x(r_{B_{TT}}) = v_x(q_B)$. Combining this to (17), we deduce that by definition (9) of the relation R, we have $(q_B, r_{B_{TT}}) \in R$.

Case 3: $\beta \in \{ok_\alpha, p_\alpha\}$

By Definition 6, there is a transition $l_{TT} \xrightarrow{\beta} l'_{TT}$ in B_{TT}, where β is labelled either by the port ok_α or p_α. The transition p_α changes only location of the TTCC component (from $read$ to $send$ location). Whereas the transition ok_α changes the location of the TTCC component (from try to $send$) and the location of the CRP (from r_α to w_α). In both cases, locations of other components are intact. We denote G^*, TC^* and F^* respectively the guard, timing constraint and update function of the transition β. Therefore, we have:

$$ \begin{aligned} q_{B_{TT}} &= \big((l^B_{TT}, l^{TTCC}_{TT}(q_{B_{TT}}), l^{CRP}_{TT}(q_{B_{TT}})),\\ &\qquad v_x(q_{B_{TT}}), v_c(q_{B_{TT}})\big)\,,\\ r_{B_{TT}} &= \big((l'^B_{TT}, l'^{TTCC}_{TT}(r_{B_{TT}}), l'^{CRP}_{TT}(r_{B_{TT}})),\\ &\qquad v'_x(r_{B_{TT}}), v_c(r_{B_{TT}})\big)\,,\\ G^*(v_x(q_{B_{TT}})) &= \mathit{True}\,,\\ TC^*(v_c(q_{B_{TT}})) &= \mathit{True}\,,\\ v_c(r_{B_{TT}}) &= v_c(q_{B_{TT}})\\ v_x(r_{B_{TT}}) &= F^*(v_x(q_{B_{TT}}))\,,\,, \end{aligned} \tag{19} $$

In the before last equality of (19), we have $v_c(r_{B_{TT}}) = v_c(q_{B_{TT}})$ since transition is instantaneous. For the last equality of (19), notice that, F^* operates only on variables that are local to the TTCC component. Therefore this function does not update variables of the components B^{TT}_i that are common with the model B. Therefore the execution of this update function does not change the valuation v^*_x. Thus, we have:

$$ v^*_x(r_{B_{TT}}) = v^*_x(q_{B_{TT}})\,. \tag{20} $$

By definition of the transformation rules (Rule 54, Rule 55 and Rule 56), after executing this β transition, the TTCC component reaches the place $send$ and the CRP component reaches back the place $wait$. The component B^{TT}_i does not change its location. Thus, we have :

$$ {l'}^B_{TT} = l^B_{TT}\,. \tag{21} $$

By construction (9) of R, we have $q_B = \big(l, v_x(q_B), v_c(q_B)\big)$, such that

$$ \begin{aligned} l^B_{TT} &\in \{l_i, \perp^{l_i}_{p_i}\}^n\,,\\ v_c(q_B) &= v_c(q_{B_{TT}})\,,\\ v_x(q_B) &= v^*_x(q_{B_{TT}})\,. \end{aligned} \tag{22} $$

Combining (19), (20), (21) and (22), we obtain that $v_c(r_{B_{TT}}) = v_c(q_B)$, $v^*_x(r_{B_{TT}}) = v_x(q_B)$ and ${l'}^B_{TT} = l^B_{TT} \in \{l_i, \perp^{l_i}_{p_i}\}^n$. Thus, we deduce that by definition (9) of the relation R, we have $(q_B, r_{B_{TT}}) \in R$.

Case 4: β is a delay step labelled by $\delta \in \mathbb{R}_+$.

By Definition 6, there is a tpc constraint on location l_{TT} in B_{TT}, $tpc(l_{TT}) = (c^g \le v)$. That is the tpc condition of each location of each component of the B_{TT} model that is composing the global location l_{TT} must satisfy this same condition. Therefore, we have:

$$ \begin{aligned} q_{B_{TT}} &= \big(l_{TT}, v_x(q_{B_{TT}}), v_c(q_{B_{TT}})\big)\,,\\ r_{B_{TT}} &= \big(l_{TT}, v_x(r_{B_{TT}}), v_c(r_{B_{TT}})\big)\,,\\ v_x(r_{B_{TT}}) &= v_x(q_{B_{TT}}),\\ v_c(r_{B_{TT}}) &= v_c(q_{B_{TT}}) + \delta\,, v_c(q_{B_{TT}}) + \delta \le v\,. \end{aligned} \tag{23} $$

Note that, by construction of the transformation (Rule 54, Rule 55), this delay transition is only possible if at least one conflicting TTCC component is not occupying the $send$ place, i.e. $l^{TTCC^C}_{TT} \neq \{send\}^k$. After executing this β transition, the TTCC component does not change the global place nor the variables valuation, only the clock valuation is augmented by δ. Thus, we have :

$$ {l'}^B_{TT} = l\,. \tag{24} $$

By construction of the definition (9) of R, we have $q_B = \big(l, v_x(q_B), v_c(q_B)\big)$, such that

$$v_c(q_B) = v_c(q_{B_{TT}}) \quad \text{and} \quad v_x(q_B) = v_x^*(q_{B_{TT}})\,. \tag{25}$$

By definition of the transformation (see Rule 54, Rule 55), the tpc constraints of the TTCC component is the conjunction of time progress conditions received in the offers from participating components. Thus there exist a transition $q_B \xrightarrow[B]{\delta} r_B$ in B where $r_B = (l, v_x(r_B), v_c(r_B))$ such that:

$$v_c(q_B) = v_c(r_B) + \delta \quad \text{and} \quad v_x(q_B) = v_x(r_B)\,. \tag{26}$$

Combining (23), (25) and (26), we obtain that $v_c(r_{B_{TT}}) = v_c(r_B)$ and $v_x^*(r_{B_{TT}}) = v_x(r_B)$. Combining this to (24), we deduce that by definition (9) of the relation R, we have $(r_B, r_{B_{TT}}) \in R$.

(iii) Let $(q_B, q_{B_{TT}}) \in R$ such that $q_B \xrightarrow[B]{\alpha} r_B$. If $\beta(\alpha) \neq \emptyset \wedge q_B \xrightarrow[B]{\alpha} r_B$, then by definition (8) of the relation β, $\alpha \in \gamma$ and can be either an internal ($\alpha \in A_I$) or an external interaction ($\alpha \in A_E$).

Case 1: $\alpha \in \gamma \cap A_I$**.**

By Definition 6, there is a transition $l \xrightarrow{\alpha} l'$ in B, where α is guarded by G^*, the timing constraint TC^* and having as transfer function F^*, such that:

$$\begin{aligned} &q_B = \big(l, v_x(q_B), v_c(q_B)\big)\,, \quad r_B = \big(l', v_x(r_B), v_c(r_B)\big)\,,\\ &\quad TC^*(v_c(q_B)) = \mathit{True}, \quad G^*(v_x(q_B)) = \mathit{True},\\ &\quad v_x(r_B) = F^*(v_x(q_B)), \quad \text{and} \quad v_c(r_B) = v_c(q_B)\,, \end{aligned} \tag{27}$$

where the update function $F* = f_i \circ \cdots \circ f_j \circ F_\alpha$, where f_i corresponds to the update function of the transition labelled by port $p_i \in P_\alpha$ in the component $B_i \in \mathit{comp}(\alpha)$. By construction (9) of R, we have $q_{B_{TT}} = \big(l_{TT}, v_x(q_{B_{TT}}), v_c(q_{B_{TT}})\big)$, such that

$$v_c(q_B) = v_c^*(q_{B_{TT}}) \quad \text{and} \quad v_x(q_B) = v_x^*(q_{B_{TT}})\,. \tag{28}$$

By definition of the transformation (Rule 54, Rule 55 and Rule 51), this interaction remains intact in the obtained B_{TT} model. Therefore, by Definition 6, we also have $q_{B_{TT}} \xrightarrow[B_{TT}]{\alpha} r_{B_{TT}}$, where $r_{B_{TT}} = \big(l'_{TT}, v_x(r_{B_{TT}}), v_c(r_{B_{TT}})\big)$ such that:

$$\begin{aligned} &{l'}_{TT}^{B} = l'\,,\\ &v_c(r_{B_{TT}}) = v_c(q_{B_{TT}})\,,\\ &v_x^*(r_{B_{TT}}) = F^*\big(v_x^*(q_{B_{TT}})\big)\,. \end{aligned} \tag{29}$$

In the second equality of (29), we have $v_c(r_{B_{TT}}) = v_c(q_{B_{TT}})$ since transition α is instantaneous. For the last equality of (29), notice that, v_x^* operates only on common variables between models B and B_{TT}.
Combining (27), (28) and (29) we obtain that l_{TT} satisfies ${l'}_{TT}^{B} = l'$, $v_c^*(r_{B_{TT}}) = v_c(r_B)$ and $v_x^*(r_{B_{TT}}) = v_x(r_B)$. Thus, we have $q_{B_{TT}} \xrightarrow[B_{TT}]{\alpha} r_{B_{TT}}$ such that $(\alpha, \alpha) \in \beta$ since $\alpha \in \gamma \cap A_I$. By definition (9) of the relation R, we obtain $(r_B, r_{B_{TT}}) \in R$.

Case 2: $\alpha \in \gamma \cap A_E$**.**

By Definition 6, there is a transition $l \xrightarrow{\alpha} l'$ in B, where α is guarded by G^*, the timing constraint TC and having as transfer function F^*, such that:

$$\begin{aligned} &q_B = \big(l, v_x(q_B), v_c(q_B)\big)\,, \quad r_B = \big(l', v_x(r_B), v_c(r_B)\big)\,,\\ &\quad TC^*(v_c(q_B)) = \mathit{True}, \quad G^*(v_x(q_B)) = \mathit{True},\\ &\quad v_x(r_B) = F^*(v_x(q_B)), \quad \text{and} \quad v_c(r_B) = v_c(q_B)\,, \end{aligned} \tag{30}$$

where the update function $F* = f_i \circ \cdots \circ f_j \circ F_\alpha$, where f_i corresponds to the update function of the transition labelled by port $p_i \in P_\alpha$ in the component $B_i \in \mathit{comp}(\alpha)$. By construction (9) of R, we have $q_{B_{TT}} = \big(l_{TT}, v_x(q_{B_{TT}}), v_c(q_{B_{TT}})\big)$, such that

$$v_c(q_B) = v_c^*(q_{B_{TT}}) \quad \text{and} \quad v_x(q_B) = v_x^*(q_{B_{TT}})\,. \tag{31}$$

By definition of the transformation (Rule 54, Rule 55 and Rule 51), the interaction α of the original model B is held by a dedicated TTCC component that we denote here $TTCC_\alpha$ in the obtained B_{TT} model. It may be mapped to the following successive transitions in the B_{TT} model:

- If the component l_{TT}^B of the global place l_{TT} contains a place $l_i^{TT} = \perp_{p_i}^{l_i}$, where $B_i \in \mathit{comp}(\alpha)$ and $p_i \in P_\alpha$, then a sending offer interaction may be enabled, note that by definition of β, this interaction is a β transition. If the component l_{TT}^B of the global place l_{TT} is equal to l (i.e. $l_{TT}^B = (l_1, .., l_n)$), no offer transition is enabled.
- Once all offers of components $B_i \in \mathit{comp}(\alpha)$ are send to $TTCC_\alpha$, then this latter reaches the place *read*. If initially, α is not conflicting, then from the reached global location, after sending offers, the transition labelled by the unary interaction p_α is enabled. This transition has the guard G^*, the timing constraint TC^* and executes the function F^*. Note that by definition of β, $\beta(p_\alpha) = \emptyset$. If α is initially a conflicting interaction, then from the reached global location, after sending offers, the enabled transition is the $rsv\alpha$ interaction. This interactions has the guard G^* and the timing constraint TC^*. By definition of β, $\beta(rsv_\alpha) = \emptyset$, it is then a β transition. From the reached location by the rsv_α interaction, two interactions are possible, $fail_\alpha$ or ok_α. $\beta(fail_\alpha) = \emptyset$ and $\beta(ok_\alpha) = \emptyset$. If the $fail_\alpha$ interaction is enabled then the $TTCC_\alpha$ component is reaching back the state enabling again the $rsv\alpha$ interaction until the $ok\alpha$ is enabled. From this reached global location a loop of rsv_α and $fail_\alpha$ may be enabled before the ok_α interaction is enabled. This latter reaches a state where the $TTCC_\alpha$ is in place *send*. The ok_α as well as the p_α transition applies the update function F^* to the local variables that are local to the TTCC. Note that these variables are not concerned by the valuation v_x^*.
- Note that after the previously executed interaction the components $B_i \in \mathit{comp}(\alpha)$ do not change their locations. The $TTCC_\alpha$ component reaches the *send* location. From this new reached global state, the notification interaction is enabled. It relates the port p_s^α of the $TTCC_\alpha$ to ports p_i of components B_i, such that $p_i \in P_\alpha$. Note that $\beta(p_s^\alpha) \neq \emptyset$. This notification interaction updates variables of components B_i according to their copies in the component $TTCC_\alpha$. Note that these copies have been transformed by F^* in the previous β transition. The reached location of

the notification interaction in a component B_i is l'_i or $\bot^{l'_i}_{p'_i}$, where $l'_i \xrightarrow{p'_i}$.

Notice that in the previously cited cases of possible interactions, we consider only β interactions in which the $TTCC\alpha$ participates. For clarity reasons, we do not detail different other possible β transitions involving other TTCC components and potential offer sending requests. Not considering them, does not invalidate this proof since they always satisfy the property $l^B_{TT} \in \{l_i, \bot^{l_i}_{p_i}\}^n$, are instantaneous and do not hold any update function (i.e. they do not impact the location property, nor the clock and variables valuations).

Therefore, by Definition 6, we have:

$$q_{B_{TT}} \xrightarrow[B_{TT}]{\beta^*} q'_{B_{TT}} \xrightarrow[B_{TT}]{p^\alpha_s} r_{B_{TT}}\,,$$

where

$$\begin{aligned} q'_{B_{TT}} &= \big((l^B_{TT}, l^{TTCC}_{TT}(q'_{B_{TT}}), l'^{CRP}_{TT}(q'_{B_{TT}})), v_x(q'_{B_{TT}}),\\ &\quad v_c(q'_{B_{TT}})\big)\,,\\ r_{B_{TT}} &= \big((l'^B_{TT}, l^{TTCC}_{TT}(r_{B_{TT}}), l^{CRP}_{TT}(r_{B_{TT}})), v_x(r_{B_{TT}}),\\ &\quad v_c(q'_{B_{TT}})\big)\,, \end{aligned}$$

with

$$\begin{aligned} &l'^B_{TT} \in \{l'_i, \bot^{l'_i}_{p'_i}\}^n\,,\\ &v_c(r_{B_{TT}}) = v_c(q'_{B_{TT}}) = v_c(q_{B_{TT}})\,,\\ &v^*_x(q'_{B_{TT}}) = v^*_x(q_{B_{TT}})\,,\\ &v^*_x(r_{B_{TT}}) = F^*\big(v^*_x(q'_{B_{TT}})\big)\,, \end{aligned} \tag{32}$$

For the last equality of (32), notice that, v^*_x operates only on common variables between models B and B_{TT}. And F^* has been first applied to local variables of the TTCC component in the β transition preceding the p^α_s transition. These variables are not concerned by the v^*_x valuation, thus, the equality $v^*_x(q'_{B_{TT}}) = v^*_x(q_{B_{TT}})$. The transition p^α_s copies values of TTCC variables to those of B_i components. Thus the function F^* is indirectly applied to variables of B_i. Which explains the equality $v^*_x(r_{B_{TT}}) = F^*\big(v^*_x(q'_{B_{TT}})\big)$.

Combining (30), (31) and (32), we obtain that l'_{TT} satisfies $l'^B_{TT} \in \{l'_i, \bot^{l'_i}_{p'_i}\}^n$, $v^*_c(r_{B_{TT}}) = v_c(r_B)$ and $v^*_x(r_{B_{TT}}) = v_x(r_B)$. Thus, we have $q_{B_{TT}} \xrightarrow[B_{TT}]{\beta^* p^\alpha_s} r_{B_{TT}}$ such that $(\alpha, p^\alpha_s) \in \beta$. By definition (9) of the relation R, we obtain $(r_B, r_{B_{TT}}) \in R$.

(iv) Let $(q_B, q_{B_{TT}}) \in R$ such that $q_{B_{TT}} \xrightarrow[B_{TT}]{\alpha_{TT}} r_{B_{TT}}$. If $\beta^{-1}(\alpha_{TT}) \neq \emptyset \wedge q_{B_{TT}} \xrightarrow[B_{TT}]{\alpha_{TT}} r_{B_{TT}}$, then by definition (8) of the relation β,

$$\alpha_{TT} \in (\gamma \cap A_I) \cup \{p^\alpha_s \in \gamma_{TT} \mid \alpha \in \gamma \cap A_E\}$$

Case 1: $\alpha_{TT} = \alpha \in \gamma \cap A_I$.

By Definition 6, there is a transition $l_{TT} \xrightarrow{\alpha_{TT}} l'_{TT}$ in B_{TT}, where the transition α_{TT} has a guard G^*, a timing constraint TC^* and an update function F^*, such that:

$$\begin{aligned} q_{B_{TT}} &= \big((l, l^{TTCC}_{TT}(q_{B_{TT}}), l^{CRP}_{TT}(q_{B_{TT}})),\\ &\quad v_x(q_{B_{TT}}), v_c(q_{B_{TT}})\big)\,,\\ r_{B_{TT}} &= \big(l', l'^{TTCC}_{TT}(r_{B_{TT}}), l'^{CRP}_{TT}(r_{B_{TT}})\big)\\ &\quad v_x(r_{B_{TT}}), v_c(r_{B_{TT}})\big)\,,\\ G^*(v_x(q_{B_{TT}})) &= \mathit{True}\,,\\ TC^*(v_c(q_{B_{TT}})) &= \mathit{True}\,,\\ v_x(r_{B_{TT}}) &= F^*(v_x(q_{B_{TT}})\,,\\ v_c(r_{B_{TT}}) &= v_c(q_{B_{TT}})\,. \end{aligned} \tag{33}$$

By definition of the transformation (cf. Rule 55 and Rule 51), the transition $\alpha_{TT} = \alpha$ is exactly the same as in the model B which corresponds to the following transition $l \xrightarrow{\alpha} l'$ in B, which is guarded by G^*, TC^* and has the update function F^*.

By construction (9) of R, we have $q_B = \big(l, v_x(q_B), v_c(q_B)\big)$, such that

$$v_c(q_B) = v_c(q_{B_{TT}}) \quad \text{and} \quad v_x(q_B) = v^*_x(q_{B_{TT}})\,. \tag{34}$$

Therefore, By Definition 6, we also have $q_B \xrightarrow[B]{\alpha} r_B$,, where

$$r_B = \big(l', v_x(r_B), v_c(r_B)\big)\,,$$

with

$$\begin{aligned} G^*(v_x(q_B)) &= \mathit{True}\,,\\ TC^*(v_c(q_B)) &= \mathit{True}\,,\\ v_c(r_B) &= v_c(q_B)\,,\\ v_x(r_B) &= F^*(v_x(q_B))\,. \end{aligned} \tag{35}$$

Combining (33), (34) and (35), we obtain that l'_{TT} satisfies $l'^B_{TT} = l' \in \{l_i, \bot^{l_i}_{p_i}\}^n$, $v_c(r_{B_{TT}}) = v_c(r_B)$ and $v^*_x(r_{B_{TT}}) = v_x(r_B)$. Thus, we have $q_B \xrightarrow[B]{\alpha} r_B$ and, by definition (9) of the relation R, $(r_B, r_{B_{TT}}) \in R$.

Case 2: $\alpha_{TT} = p^\alpha_s, \alpha \in \gamma \cap A_E$.

By Definition 6, there is a transition $l_{TT} \xrightarrow{\alpha_{TT}} l'_{TT}$ in B_{TT}. The transition α_{TT} has no guard.

By construction of the transformation (cf. Rule 54, Rule 55 and Rule 51), this α_{TT} transition is always preceded by a β transition consisting in p_α if α is not conflicting and in ok_α if α is conflicting. These latter execute an update function F^* that updates variables local to the TTCC component. These variables are local copies of variables of B_i. When receiving offers, values of variables of the TTCC component are the same as their remote copies in B_i components. And then, they are updated by using the function F^* of transition ok_α or p_α.

The notification transition is not guarded and have an update function which copies values of local variables of the TTCC to their corresponding copies in the participating B_i components. Therefore the function F^* is indirectly applied to variables of B_i components. These variables are concerned by the v^*_x valuation.

Note that this α_{TT} transition, changes the location of the TTCC component to its initial *wait* location and allows to reach location l'_i or $\bot^{l'_i}_{p'_i}$, where $l'_i \xrightarrow{p'_i}$ and $p'_i \in A_E$.

Therefore, we have $l_{TT} \xrightarrow{\alpha_{TT}} l'_{TT}$, such that:

$$\begin{aligned} q_{B_{TT}} &= \big((l^{B}_{TT}(q_{B_{TT}}), l^{TTCC}_{TT}(q_{B_{TT}}), l^{CRP}_{TT}(q_{B_{TT}}), \\ &\quad v_x(q_{B_{TT}}), v_c(q_{B_{TT}})\big)\,, \\ r_{B_{TT}} &= \big(l'^{B}_{TT}(q_{B_{TT}}), l'^{TTCC}_{TT}(r_{B_{TT}}), l'^{CRP}_{TT}(r_{B_{TT}}), \\ &\quad v_x(r_{B_{TT}}), v_c(r_{B_{TT}})\big)\,, \\ v^*_x(r_{B_{TT}}) &= F^*(v^*_x(q_{B_{TT}}))\,, \\ v_c(r_{B_{TT}}) &= v_c(q_{B_{TT}})\,, \end{aligned} \tag{36}$$

such that

$$l'^{B}_{TT} \in \{l'_i, \perp^{l'_i}_{p'_i}\}^n\,. \tag{37}$$

By definition of the transformation (cf. Rule 54, Rule 55 and Rule 51), there exist a corresponding transition $l \xrightarrow{\alpha} l'$ in B, which is having as transfer function F^*.
By construction (9) of R, we have $q_B = \big(l, v_x(q_B), v_c(q_B)\big)$, such that

$$\begin{aligned} l^{B}_{TT}(q_{B_{TT}}) &\in \{l_i, \perp^{l_i}_{p_i}\}^n\,, \\ v_c(q_B) &= v_c(q_{B_{TT}})\,, \\ v_x(q_B) &= v^*_x(q_{B_{TT}})\,. \end{aligned} \tag{38}$$

Therefore, By Definition 6, we also have $q_B \xrightarrow[B]{\alpha} r_B$,, where

$$r_B = \big(l', v_x(r_B), v_c(r_B)\big)\,,$$

with

$$\begin{aligned} v_c(r_B) &= v_c(q_B)\,, \\ v_x(r_B) &= F^*(v_x(q_B))\,. \end{aligned} \tag{39}$$

Combining (36), (37), (38) and (39), we obtain that l'_{TT} satisfies $l'^{B}_{TT} \in l'_i, \perp^{l'_i}_{p'_i}{}^n$, $v_c(r_{B_{TT}}) = v_c(r_B)$ and $v^*_x(r_{B_{TT}}) = v_x(r_B)$. Thus, we have $q_B \xrightarrow[B]{\alpha} r_B$ and, by definition (9) of the relation R, $(r_B, r_{B_{TT}}) \in R$.