Compliance and subtyping in timed session types

Massimo Bartoletti, Tiziana Cimoli, Maurizio Murgia,
Alessandro Sebastian Podda, and Livio Pompianu

Università degli Studi di Cagliari, Italy

Abstract. We propose an extension of binary session types, to formalise timed communication protocols between two participants at the endpoints of a session. We introduce a decidable compliance relation, which generalises to the timed setting the usual progress-based notion of compliance between untimed session types. We then show a sound and complete technique to decide when a timed session type admits a compliant one, and if so, to construct the least session type compliant with a given one, according to the subtyping preorder induced by compliance. Decidability of subtyping follows from these results. We exploit our theory to design and implement a message-oriented middleware, where distributed modules with compliant protocols can be dynamically composed, and their communications monitored, so to guarantee safe interactions.

1 Introduction

Session types are formal descriptions of interaction protocols involving two or more participants over a network [19,25]. They can be used to specify the behavioural interface of a service or a component, and to statically check through a (session-)type system that this conforms to its implementation, so enabling compositional verification of distributed applications. Session types support formal definitions of compatibility or *compliance* (when two or more session types, composed together, behave correctly), and of substitutability or *subtyping* (when a service can be safely replaced by another one, while preserving the interaction capabilities with the context). Since these notions are often decidable and computationally tractable (for synchronous session types), or safely approximable (for asynchronous ones), session typing is becoming a particularly attractive approach to the problem of correctly designing distributed applications. This is witnessed by a steady flow of foundational studies [16,10,15] and of tools [12,26] based on them in the last few years.

In the simplest setting, session types are terms of a process algebra featuring a selection construct (an *internal choice* among a set of branches), a branching construct (an *external choice* offered to the environment), and recursion. In this basic form, session types cannot faithfully capture a natural and relevant aspect of interaction protocols, i.e., the timing constraints among the communication actions. While formal methods for time have been studied for at least a couple of decades, they have approached the realm of session types very recently [9,24].

However, these approaches introduce time into an already sophisticated framework, featuring multiparty session types with asynchronous communication (via unbounded buffers). While on the one hand this has the advantage of extending to the timed setting type techniques which enable compositional verification [20], on the other hand it seems that some of the key notions of the untimed setting (e.g., compliance, duality) have not been explored yet in the timed case.

We think that studying timed session types in a basic setting (synchronous communication between two endpoints, as in the seminal untimed version) is worthy of attention. From a theoretical point of view, the objective is to lift to the timed case some decidability results, like those of compliance and subtyping. Some intriguing problems arise: unlike in the untimed case, a timed session type not always admits a compliant; hence, besides deciding if two session types *are compliant*, it becomes a relevant problem whether a session type *has a compliant*. From a more practical perspective, decision procedures for timed session types, like those for compliance and for dynamic verification, enable the implementation of programming tools and infrastructures for the development of safe communication-oriented distributed applications.

Contributions. In this paper we introduce a theory of binary timed session types (TSTs), and we explore its viability as a foundation for programming tools to leverage the complexity of developing distributed applications.

We start in Section 2 by giving the syntax and semantics of TSTs. E.g., we describe as the following TST the contract of a service taking as input a zip code, and then either providing as output the current weather, or aborting:

$$p \;=\; \texttt{?zip}\{x\}.\,(\texttt{!weather}\{5 < x < 10\} \oplus \texttt{!abort}\{x < 1\})$$

The prefix $\texttt{?zip}\{x\}$ states that the service can receive a `zip` code, and then reset a clock x. The continuation is an *internal choice* between two outputs: either the service sends `weather` in a time window of $(5, 10)$ time units, or it will `abort` the protocol within 1 time unit.

The semantics of TSTs is a conservative extension of the synchronous semantics of untimed session types [4], adding *clock valuations* to associate each clock with a positive real. We also extend to the timed setting the standard semantic notion of *compliance*, which relates two session types whenever they enjoy progress until reaching success. For instance, p above is *not* compliant with:

$$q \;=\; \texttt{!zip}\{y\}.\,(\texttt{?weather}\{y < 7\} + \texttt{?abort}\{y < 5\})$$

because q is available to receive `weather` until 7 time units since it has sent the `zip` code, while p can choose to send `weather` until 10 time units (note that p and q, cleaned from all time annotations, are compliant in the untimed setting).

Despite the semantics of TSTs being infinite-state (while it is finite-state in the untimed case), we develop a sound and complete decision procedure for verifying compliance (Theorem 1). To do that, we reduce this problem to that of model-checking deadlock freedom in timed automata [2], which is decidable, and we implement our technique using the Uppaal model checker [7].

Another difference from the untimed case is that not every TST admits a compliant (while in the untimed case, a session type is always compliant to its syntactic dual). For instance, consider the client contract:

$$q' \;=\; !\mathtt{zip}\{y < 10\}.\,(?\mathtt{weather}\{y < 7\} + ?\mathtt{abort}\{y < 5\})$$

No service can be compliant with q', because if q' sends the `zip` code, e.g., at time 8, one cannot send `weather` or `abort` in the given time constraints. We develop a procedure to detect whether a TST admits a compliant. This takes the form of a kind system which associates, to each p, a set of clock valuations under which p admits a compliant. The kind system is sound and complete (Theorems 4 and 5), and kind inference is decidable (Theorem 3), so summing up we have a (sound and complete) decision procedure for the existence of compliant. When p admits a compliant, by exploiting the kind system we can construct the *greatest* TST compliant with p (Theorem 7), according to the semantic subtyping relation [4]. Decidability of subtyping follows from that of compliance and kind inference. This provides us with an effective way of checking whether a service with type p can be replaced by one with a subtype p' of p, guaranteeing that all the services which interacted correctly with the old one will do the same with the new one.

In Section 4 we address the problem of dynamically monitoring interactions regulated by TSTs. To do that, we will provide TSTs with a *monitoring semantics*, which detects when a participant is not respecting its TST. This semantics enjoys some desirable properties: it is deterministic, and it guarantees that in each state of an interaction, either we have reached success, or someone is in charge of a move, or not respecting its TST. We then exploit all the theoretical results discussed above, to discuss the design and implementation of a message-oriented middleware which uses TSTs to enable and regulate the interaction of distributed services. This infrastructure pursues the bottom-up approach to service composition: it allows services to advertise contracts (in the form of TSTs); all the advertised TSTs are collected by a *broker*, which finds pairs of compliant TSTs, and creates *sessions* between the respective services. These can then start interacting, by doing the actions prescribed by their TSTs (or even by choosing not to do so). In a system of honest services, compliance between TSTs ensures progress of the whole system; in any case, dynamic verification of all the exchanged messages guarantees safe executions.

Due to space constraints, the proofs of our statements, additional examples, as well as some tools related to the middleware, are available in [5].

2 Timed session types

We introduce binary timed session types (TSTs), by giving their syntax and semantics, and by defining a compliance relation between them. The main result of this section is Theorem 1, which states that compliance is decidable.

Syntax. Let $\mathtt{A}$ be a set of *actions*, ranged over by $\mathtt{a}, \mathtt{b}, \ldots$. We denote with $\mathtt{A}^!$ the set $\{!\mathtt{a} \mid \mathtt{a} \in \mathtt{A}\}$ of *output actions*, with $\mathtt{A}^?$ the set $\{?\mathtt{a} \mid \mathtt{a} \in \mathtt{A}\}$ of *input actions*, and with $\mathtt{L} = \mathtt{A}^! \cup \mathtt{A}^?$ the set of *branch labels*, ranged over by $\ell, \ell', \ldots$.

We use $\delta, \delta', \ldots$ to range over the set $\mathbb{R}_{\geq 0}$ of positive real numbers including zero, and $d, d', \ldots$ to range over $\mathbb{N}$. Let $\mathbb{C}$ be a set of *clocks*, namely variables in $\mathbb{R}_{\geq 0}$, ranged over by $t, t', \ldots$. We use $R, T, \ldots \subseteq \mathbb{C}$ to range over sets of clocks.

Definition 1 (Guards). *The set $\mathcal{G}_{\mathbb{C}}$ of* guards *over clocks $\mathbb{C}$ is defined as:*

$$g \;::=\; \mathtt{true} \;\mid\; \neg g \;\mid\; g \wedge g \;\mid\; t \circ d \;\mid\; t - t' \circ d \quad (\textit{where } \circ \in \{<, \leq, =, \geq, >\})$$

Definition 2 below introduces the syntax of TSTs. A TST p models the behaviour of a single participant involved in an interaction. TSTs are terms of a process algebra featuring the *success* state $\mathbf{1}$, *internal choice* $\bigoplus_{i \in I} \mathtt{!a}_i\{g_i, R_i\}\,.\,p_i$, *external choice* $\sum_{i \in I} \mathtt{?a}_i\{g_i, R_i\}\,.\,p_i$, and recursion $\mathrm{rec}\, X\,.\,p$.

To give some intuition, we consider two participants, Alice (A) and Bob (B), which want to interact. Alice advertises an internal choice $\bigoplus_i \mathtt{!a}_i\{g_i, R_i\}\,.\,p_i$ when she wants to do one of the outputs $\mathtt{!a}_i$ in a time window where g_i is true; further, the clocks in R_i will be reset after the output is performed. The meaning of an external choice $\sum_i \mathtt{?a}_i\{g_i, R_i\}\,.\,q_i$ (advertised, say, by B) is somehow dual: B is saying that he is available to receive each message $\mathtt{a}_i$ in *any instant* within the time window defined by g_i (and the clocks in R_i will be reset after the input).

Definition 2 (Timed session types). Timed session types $p, q, \ldots$ *are terms of the following grammar:*

$$p \;::=\; \mathbf{1} \;\mid\; \bigoplus_{i \in I} \mathtt{!a}_i\{g_i, R_i\}\,.\,p_i \;\mid\; \sum_{i \in I} \mathtt{?a}_i\{g_i, R_i\}\,.\,p_i \;\mid\; \mathrm{rec}\, X\,.\,p \;\mid\; X$$

where (i) the set I is finite and non-empty, (ii) the actions in internal/external choices are pairwise distinct, (iii) recursion is guarded. Unless stated otherwise, we consider TSTs up-to unfolding of recursion. A TST is closed *when it has no recursion variables. If $q = \bigoplus_{i \in I} \mathtt{!a}_i\{g_i, R_i\}\,.\,p_i$ and $0 \notin I$, we write $\mathtt{!a}_0.p_0 \oplus q$ for $\bigoplus_{i \in I \cup \{0\}} \mathtt{!a}_i\{g_i, R_i\}\,.\,p_i$ (the same for external choices). True guards, empty resets, and trailing occurrences of the success state can be omitted.*

Example 1. Along the lines of PayPal User Agreement [1], we specify the protection policy for buyers of a simple on-line payment platform, called PayNow (see [5] for the full version). PayNow helps customers in on-line purchasing, providing protection against misbehaviours. In case a buyer has not received what he has paid for, he can open a dispute within 180 days from the date the buyer made the payment. After opening of the dispute, the buyer and the seller may try to come to an agreement. If this is not the case, within 20 days, the buyer can escalate the dispute to a claim. However, the buyer must wait at least 7 days from the date of payment to escalate a dispute. Upon not reaching an agreement, if still the buyer does not escalate the dispute to a claim within 20 days, the dispute is considered aborted. During a claim procedure, PayNow will ask the buyer to provide documentation to certify the payment, within 3 days of the date the dispute was escalated to a claim. After that, the payment will be refunded

within 7 days. The contract of PayNow is described by the following TST p:

$$\begin{aligned} p &= \texttt{?pay}\{t_{pay}\}.\,(\texttt{?ok} + \texttt{?dispute}\{t_{pay} < 180, t_d\}.\,p') && \text{where} \\ p' &= \texttt{?ok}\{t_d < 20\} + \\ &\quad \texttt{?claim}\{t_d < 20 \wedge t_{pay} > 7, t_c\}.\texttt{?rcpt}\{t_c < 3, t_c\}.\texttt{!refund}\{t_c < 7\} + \\ &\quad \texttt{?abort} \end{aligned}$$

Semantics. To define the behaviour of TSTs we use *clock valuations*, which associate each clock with its value. The state of the interaction between two TSTs is described by a *configuration* $(p, \nu) \mid (q, \eta)$, where the clock valuations ν and η record (keeping the same pace) the time of the clocks in p and q, respectively. The dynamics of the interaction is formalised as a transition relation between configurations (Definition 5). This relation describes all and only the *correct* interactions: for instance, we do not allow time passing to make unsatisfiable all the guards in an internal choice, since doing so would prevent a participant from respecting her protocol. In Section 4 we will study another semantics of TSTs, which can also describe the behaviour of dishonest participants who do not respect their protocols.

We denote with $\mathbb{V} = \mathbb{C} \to \mathbb{R}_{\geq 0}$ the set of clock valuations (ranged over by $\nu, \eta, \ldots$), and with ν_0 the valuation mapping each clock to zero. We write $\nu + \delta$ for the valuation which increases ν by δ, i.e., $(\nu + \delta)(t) = \nu(t) + \delta$ for all $t \in \mathbb{C}$. For a set $R \subseteq \mathbb{C}$, we write $\nu[R]$ for the *reset* of the clocks in R, i.e., $\nu[R](t) = 0$ if $t \in R$, and $\nu[R](t) = \nu(t)$ otherwise.

Definition 3 (Semantics of guards). *For all guards g, we define the set of clock valuations $[\![g]\!]$ inductively as follows, where $\circ \in \{<, \leq, =, \geq, >\}$:*

$$[\![\texttt{true}]\!] = \mathbb{V} \qquad [\![\neg g]\!] = \mathbb{V} \setminus [\![g]\!] \qquad [\![g_1 \wedge g_2]\!] = [\![g_1]\!] \cap [\![g_2]\!]$$
$$[\![t \circ d]\!] = \{\nu \mid \nu(t) \circ d\} \qquad [\![t - t' \circ d]\!] = \{\nu \mid \nu(t) - \nu(t') \circ d\}$$

Before defining the semantics of TSTs, we recall from [8] some basic operations on *sets* of clock valuations (ranged over by $\mathcal{K}, \mathcal{K}', \ldots \subseteq \mathbb{V}$).

Definition 4 (Past and inverse reset). *For all sets $\mathcal{K}$ of clock valuations, the set of clock valuations $\downarrow \mathcal{K}$ (the* past *of $\mathcal{K}$) and $\mathcal{K}[T]^{-1}$ (the inverse reset of $\mathcal{K}$) are defined as: $\downarrow \mathcal{K} = \{\nu \mid \exists \delta \geq 0 : \nu + \delta \in \mathcal{K}\}$, $\mathcal{K}[T]^{-1} = \{\nu \mid \nu[T] \in \mathcal{K}\}$.*

Definition 5 (Semantics of TSTs). *A* configuration *is a term of the form $(p, \nu) \mid (q, \eta)$, where p, q are TSTs extended with* committed choices $[\texttt{!a}\{g, R\}]\,p$. *The semantics of TSTs is defined as a labelled relation $\to$ over configurations, whose labels are either silent actions τ, delays δ, or branch labels.*

We now comment the rules in Figure 1. The first four rules are auxiliary, as they describe the behaviour of a TST in isolation. Rule $[\oplus]$ allows a TST to commit to the branch $\texttt{!a}$ of her internal choice, provided that the corresponding

$$(\texttt{!a}\{g, R\}.\, p \oplus p',\ \nu) \xrightarrow{\tau} ([\texttt{!a}\{g, R\}]\, p,\ \nu) \qquad \text{if } \nu \in [\![g]\!] \qquad [\oplus]$$

$$([\texttt{!a}\{g, R\}]\, p,\ \nu) \xrightarrow{\texttt{!a}} (p,\ \nu[R]) \qquad [!]$$

$$(\texttt{?a}\{g, R\}.\, p + p',\ \nu) \xrightarrow{\texttt{?a}} (p,\ \nu[R]) \qquad \text{if } \nu \in [\![g]\!] \qquad [?]$$

$$(p,\ \nu) \xrightarrow{\delta} (p,\ \nu + \delta) \qquad \text{if } \delta > 0 \ \wedge\ \nu + \delta \in \texttt{rdy}(p) \qquad [\textsc{Del}]$$

$$\frac{(p,\nu) \xrightarrow{\tau} (p',\nu')}{(p,\nu) \mid (q,\eta) \xrightarrow{\tau} (p',\nu') \mid (q,\eta)}\ [\text{S-}\oplus] \qquad \frac{(p,\nu) \xrightarrow{\delta} (p,\nu') \quad (q,\eta) \xrightarrow{\delta} (q,\eta')}{(p,\nu) \mid (q,\eta) \xrightarrow{\delta} (p,\nu') \mid (q,\eta')}\ [\text{S-}\textsc{Del}]$$

$$\frac{(p,\nu) \xrightarrow{\texttt{!a}} (p',\nu') \quad (q,\eta) \xrightarrow{\texttt{?a}} (q',\eta')}{(p,\nu) \mid (q,\eta) \xrightarrow{\tau} (p',\nu') \mid (q',\eta')}\ [\text{S-}\tau]$$

$$\texttt{rdy}(\textstyle\bigoplus \texttt{!a}_i\{g_i, R_i\}.\, p_i) = {\downarrow}\bigcup [\![g_i]\!] \qquad \texttt{rdy}(\textstyle\sum \cdots) = \texttt{rdy}(\mathbf{1}) = \mathbb{V} \qquad \texttt{rdy}([\texttt{!a}\{g, R\}]\, p) = \emptyset$$

Fig. 1. Semantics of timed session types (symmetric rules omitted).

guard is satisfied in the clock valuation ν. This results in the term $[\texttt{!a}\{g, R\}]\, p$, which represents the fact that the endpoint has committed to branch `!a` in a specific time instant: actually, it can only fire `!a` through rule [!] (which also resets the clocks in R), while time cannot pass. Rule [?] allows an external choice to fire any of its input actions whose guard is satisfied. Rule [Del] allows time to pass; this is always possible for external choices and success term, while for an internal choice we require that at least one of the guards remains satisfiable; this is obtained through the function `rdy` in Figure 1. The last three rules deal with configurations of two TSTs. Rule [S-$\oplus$] allows a TSTs to commit in an internal choice. Rule [S-τ] is the standard synchronisation rule *à la* CCS; note that B is assumed to read a message as soon as it is sent, so A never blocks on internal choices. Rule [S-Del] allows time to pass, equally for both endpoints.

Example 2. Let $p = \texttt{!a} \oplus \texttt{!b}\{t > 2\}$, let $q = \texttt{?b}\{t > 5\}$, and consider the following computations:

$$\begin{aligned}(p,\nu_0) \mid (q,\eta_0) &\xrightarrow{7}\xrightarrow{\tau} ([\texttt{!b}\{t > 2\}]\,, \nu_0 + 7) \mid (q, \eta_0 + 7) \\ &\xrightarrow{\tau} (\mathbf{1}, \nu_0 + 7) \mid (\mathbf{1}, \eta_0 + 7) \qquad (1)\end{aligned}$$

$$(p,\nu_0) \mid (q,\eta_0) \xrightarrow{\delta}\xrightarrow{\tau} ([\texttt{!a}]\,, \nu_0 + \delta) \mid (q, \eta_0 + \delta) \qquad (2)$$

$$(p,\nu_0) \mid (q,\eta_0) \xrightarrow{3}\xrightarrow{\tau} ([\texttt{!b}\{t > 2\}]\,, \nu_0 + 3) \mid (q, \eta_0 + 3) \qquad (3)$$

The computation in (1) reaches success, while the other two computations reach the deadlock state. In (2), p commits to the choice `!a` after some delay δ; at this point, time cannot pass (because the leftmost endpoint is a committed choice), and no synchronisation is possible (because the other endpoint is not offering `?a`). In (3), p commits to `!b` after 3 time units; here, the rightmost endpoint would

offer $\texttt{?b}$, — but not in the time chosen by the leftmost endpoint. Note that, were we allowing time to pass in committed choices, then we would have obtained e.g. that $(\texttt{!b}\{t > 2\}, \nu_0) \mid (q, \eta_0)$ never reaches deadlock — contradicting our intuition that these endpoints should not be considered compliant.

Compliance. We now extend to the timed setting the standard progress-based compliance between (untimed) session types [22,11,4]. If p is compliant with q, then whenever an interaction between p and q becomes stuck, it means that both participants have reached the success state. Intuitively, when two TSTs are compliant and participants behave honestly (according to their TSTs), then the interaction will progress, until both of them reach the success state.

Definition 6 (Compliance). *We say that* $(p, \nu) \mid (q, \eta)$ *is* deadlock *whenever* (i) *it is not the case that both* p *and* q *are* $\mathbf{1}$*, and* (ii) *there is no* δ *such that* $(p, \nu + \delta) \mid (q, \eta + \delta) \xrightarrow{\tau}$*. We then write* $(p, \nu) \bowtie (q, \eta)$ *whenever:*

$$(p, \nu) \mid (q, \eta) \rightarrow^* (p', \nu') \mid (q', \eta') \quad \textit{implies} \quad (p', \nu') \mid (q', \eta') \textit{ not deadlock}$$

We say that p *and* q *are* compliant *whenever* $(p, \nu_0) \bowtie (q, \eta_0)$ *(in short,* $p \bowtie q$*).*

Example 3. Let $p = \texttt{?a}\{t < 5\}.\texttt{!b}\{t < 3\}$. We have that p is compliant with $q = \texttt{!a}\{t < 2\}.\texttt{?b}\{t < 3\}$, but it is not compliant with $q' = \texttt{!a}\{t < 5\}.\texttt{?b}\{t < 3\}$.

Example 4. Consider a customer of PayNow (see Example 1) who is willing to wait 10 days to receive the item she has paid for, but after that she will open a claim. Further, she will instantly provide PayNow with any documentation required. The customer contract is described by the following TST, which is compliant with PayNow's contract p in Example 1:

$$\texttt{!pay}\{t_{pay}\}.(\texttt{!ok}\{t_{pay} < 10\} \oplus \texttt{!dispute}\{t_{pay} = 10\}.\texttt{!claim}\{t_{pay} = 10\}.\texttt{!rcpt}\{t_{pay} = 10\}.\texttt{?refund})$$

Compliance between TSTs is somehow more liberal than the untimed notion, as it can relate terms which, when cleaned from all the time annotations, would not be compliant in the untimed case. The following example shows e.g., that a recursive internal choice can be compliant with a non-recursive external choice — which can never happen in untimed session types.

Example 5. Consider the TSTs $p = \mathrm{rec}\, X.\,(\texttt{!a} \oplus \texttt{!b}\{x \leq 1\}.\texttt{?c}.X)$, and $q = \texttt{?a} + \texttt{?b}\{y \leq 1\}.\texttt{!c}\{y > 1\}.\texttt{?a}$. We have that $p \bowtie q$. Indeed, if p chooses the output $\texttt{!a}$, then q has the corresponding input, and they both succeed; instead, if p chooses $\texttt{!b}$, then it will read $\texttt{?c}$ when $x > 1$, and so at the next loop it is forced to choose $\texttt{!a}$, since the guard of $\texttt{!b}$ has become unsatisfiable.

Definition 7 and Lemma 1 below coinductively characterise compliance between TSTs, by extending to the timed setting the coinductive compliance for untimed session types in [3]. Intuitively, an internal choice p is compliant with q when (i) q is an external choice, (ii) for each output $\texttt{!a}$ that p can fire after δ time units, there exists a corresponding input $\texttt{?a}$ that q can fire after δ time units, and (iii) their continuations are coinductively compliant. The case where p is an external choice is symmetric.

Definition 7. *We say* $\mathcal{R}$ *is a* coinductive compliance *iff* $(p,\nu)\,\mathcal{R}\,(q,\eta)$ *implies:*

1. $p = \mathbf{1} \iff q = \mathbf{1}$
2. $p = \bigoplus_{i\in I} \mathtt{!a}_i\{g_i, R_i\}\,.\,p_i \implies \nu \in \mathtt{rdy}(p) \;\wedge\; q = \sum_{j\in J} \mathtt{?a}_j\{g_j, R_j\}\,.\,q_j \;\wedge$
   $\forall \delta, i:\ \nu+\delta \in [\![g_i]\!] \implies \exists j : \mathtt{a}_i = \mathtt{a}_j \wedge \eta+\delta \in [\![g_j]\!] \wedge (p_i, \nu+\delta[R_i])\,\mathcal{R}\,(q_j, \eta+\delta[R_j])$
3. $p = \sum_{j\in J} \mathtt{?a}_j\{g_j, R_j\}\,.\,p_j \implies \eta \in \mathtt{rdy}(q) \;\wedge\; q = \bigoplus_{i\in I} \mathtt{!a}_i\{g_i, R_i\}\,.\,q_i \;\wedge$
   $\forall \delta, i:\ \eta+\delta \in [\![g_i]\!] \implies \exists j : \mathtt{a}_i = \mathtt{a}_j \wedge \nu+\delta \in [\![g_j]\!] \wedge (p_j, \nu+\delta[R_j])\,\mathcal{R}\,(q_i, \eta+\delta[R_i])$

Lemma 1. $p \bowtie q \iff \exists\, \mathcal{R}$ *coinductive compliance* $:\ (p,\nu_0)\,\mathcal{R}\,(q,\eta_0)$

The following theorem establishes decidability of compliance. To prove it, we reduce the problem of checking $p \bowtie q$ to that of model-checking deadlock freedom in a network of timed automata constructed from p and q (see [5] for details).

Theorem 1. *Compliance between TSTs is decidable.*

3 On duality and subtyping

The dual of an untimed session type is computed by simply swapping internal choices with external ones (and inputs with outputs) [10]. A naïve attempt to extend this construction to TSTs can be to swap internal with external choices, as in the untimed case, and leave guards and resets unchanged. This construction does not work as expected, as shown by the following example.

Example 6. Consider the following TSTs:

$$p_1 = \mathtt{!a}\{x \le 2\}.\,\mathtt{!b}\{x \le 1\} \qquad p_2 = \mathtt{!a}\{x \le 2\} \oplus \mathtt{!b}\{x \le 1\}.\,\mathtt{?a}\{x \le 0\}$$
$$p_3 = \mathrm{rec}\,X\,.\,\mathtt{?a}\{x \le 1 \wedge y \le 1\}.\,\mathtt{!a}\{x \le 1, \{x\}\}.\,X$$

The TST p_1 is not compliant with its naïve dual $q_1 = \mathtt{?a}\{x \le 2\}.\,\mathtt{?b}\{x \le 1\}$: even though q_1 can do the input $\mathtt{?a}$ in the required time window, p_1 cannot perform $\mathtt{!b}$ if $\mathtt{!a}$ is performed after 1 time unit. For this very reason, no TST is compliant with p_1. Note instead that $q_1 \bowtie \mathtt{!a}\{x \le 1\}.\,\mathtt{!b}\{x \le 1\}$, which is not its naïve dual. In p_2, a similar deadlock situation occurs if the $\mathtt{!b}$ branch is chosen, and so also p_2 does not admit a compliant. The reason why p_3 does not admit a compliant is more subtle: actually, p_3 can loop until the clock y reaches the value 1; after this point, the guard $y \le 1$ can no longer be satisfied, and then p_3 reaches a deadlock.

As suggested in the above example, the dual construction makes sense only for those TSTs for which a compliant exists. To this purpose, we define a procedure (more precisely, a kind system) which computes the set of clock valuations $\mathcal{K}$ (called *kinds*) such that p admits a compliant TST in all $\nu \in \mathcal{K}$. We then provide a constructive proof of its soundness, by showing a TST q compliant with p, which we call the dual of p.

We now define our kind system for TSTs.

$$\Gamma \vdash \mathbf{1} : \mathbb{V} \ \text{[T-1]} \qquad \frac{\Gamma \vdash p_i : \mathcal{K}_i \qquad \text{for } i \in I}{\Gamma \vdash \sum_{i\in I} \mathtt{?a}_i\{g_i, T_i\}\,.\,p_i : \bigcup_{i\in I} {\downarrow}\left([\![g_i]\!] \cap \mathcal{K}_i[T_i]^{-1}\right)} \ \text{[T-+]}$$

$$\frac{\Gamma \vdash p_i : \mathcal{K}_i \qquad \text{for } i \in I}{\Gamma \vdash \bigoplus_{i\in I} \mathtt{!a}_i\{g_i, T_i\}\,.\,p_i : \left(\bigcup_{i\in I} {\downarrow}[\![g_i]\!]\right) \setminus \left(\bigcup_{i\in I} {\downarrow}([\![g_i]\!] \setminus \mathcal{K}_i[T_i]^{-1})\right)} \ \text{[T-}\oplus\text{]}$$

$$\Gamma, X : \mathcal{K} \vdash X : \mathcal{K} \ \text{[T-Var]} \qquad \frac{\exists \mathcal{K}, \mathcal{K}' : \ \Gamma\{^{\mathcal{K}}/_X\} \vdash p : \mathcal{K}'}{\Gamma \vdash \mathrm{rec}\, X\,.\,p : \bigcup \{\mathcal{K} \mid \Gamma\{^{\mathcal{K}}/_X\} \vdash p : \mathcal{K}' \wedge \mathcal{K} \subseteq \mathcal{K}'\}} \ \text{[T-Rec]}$$

Fig. 2. Kind system for TSTs.

Definition 8 (Kind system). *Kind judgements $\Gamma \vdash p : \mathcal{K}$ are defined in Figure 2. where Γ is a partial function which associates kinds to recursion variables.*

Rule [T-1] says that the success TST $\mathbf{1}$ admits compliant in every ν: indeed, $\mathbf{1}$ is compliant with itself. The kind of an exernal choice is the union of the kinds of its branches (rule [T-+]), where the kind of a branch is the past of those clock valuations which satisfy both the guard and, after the reset, the kind of their continuation. Internal choices are dealt with by rule [T-⊕], which computes the difference between the union of the past of the guards and a set of error clock valuations. The error clock valuations are those which can satisfy a guard but not the kind of its continuation. Rule [T-Var] is standard. Rule [T-Rec] looks for a kind which is preserved by unfolding of recursion (hence a fixed point). In order to obtain completeness of the kind system we need the greatest fixed point.

Example 7. Recall p_2 from Example 6. We have the following kind derivation:

$$\frac{\vdash \mathbf{1} : \mathbb{V} \qquad \dfrac{\vdash \mathbf{1} : \mathbb{V}}{\vdash \mathtt{!a}\{x \le 0\} : {\downarrow}[\![x \le 0]\!] \cap \mathbb{V} = [\![x \le 0]\!]} \ \text{[T-+]}}{\vdash p_2 : \left({\downarrow}[\![x \le 2]\!] \cup {\downarrow}[\![x \le 1]\!]\right) \setminus \left({\downarrow}[\![x \le 2]\!] \setminus \mathbb{V}\right) \cup {\downarrow}[\![x \le 1]\!] \setminus [\![x \le 0]\!]\big) = \mathcal{K}} \ \text{[T-}\oplus\text{]}$$

where $\mathcal{K} = [\![(x > 1) \wedge (x \le 2)]\!]$. As noted in Example 6, intuitively p_2 has no compliant; this will be asserted by Theorem 5 below, as a consequence of the fact that $\nu_0 \notin \mathcal{K}$. However, since $\mathcal{K}$ is non-empty, Theorem 4 guarantees that there exist q and η such that $(p_2, \nu) \bowtie (q, \eta)$, for all clock valuations $\nu \in \mathcal{K}$.

The following theorem states that every TST is kindable. We stress the fact that being kindable does not imply admitting a compliant. This holds if and only if ν_0 belongs to the kind (see Theorems 4 and 5).

Theorem 2. *For all closed p, there exists some $\mathcal{K}$ such that $\vdash p : \mathcal{K}$.*

The following theorem states that the problem of determining the kind of a TST is decidable. This might seem surprising, as the cardinality of kinds is $2^{2^{\aleph_0}}$. However, the kinds constructed by our inference rules can always be represented syntactically by guards (as in Definition 1) [18].

Theorem 3. *Kind inference is decidable.*

$$
\begin{array}{rll}
\mathrm{co}_\Gamma(\mathbf{1}) = \mathbf{1} & & \\
\mathrm{co}_\Gamma\left(\sum_{i\in I} \mathtt{?a}_i\{g_i, T_i\}\,.\,p_i\right) = \bigoplus_{i\in I} \mathtt{!a}_i\{g_i \wedge \mathcal{K}_i[T_i]^{-1}, T_i\}\,.\,\mathrm{co}_\Gamma(p_i) & \text{if } \Gamma \vdash p_i : \mathcal{K}_i \\
\mathrm{co}_\Gamma\left(\bigoplus_{i\in I} \mathtt{!a}_i\{g_i, T_i\}\,.\,p_i\right) = \sum_{i\in I} \mathtt{?a}_i\{g_i, T_i\}\,.\,\mathrm{co}_\Gamma(p_i) & \\
\mathrm{co}_\Gamma(X) = X & \text{if } \Gamma(X) \text{ defined} \\
\mathrm{co}_\Gamma(\mathrm{rec}\,X\,.\,p) = \mathrm{rec}\,X\,.\,\mathrm{co}_{\Gamma\{\mathcal{K}/X\}}(p) & \text{if } \Gamma \vdash \mathrm{rec}\,X\,.\,p : \mathcal{K}
\end{array}
$$

Fig. 3. Dual of a TST.

We now define the *canonical compliant* of kindable TSTs. Roughly, we turn internal choices into external ones (without changing guards nor resets), and external into internal, changing the guards so that the kind of continuations is preserved. Decidability of this construction follows from that of kind inference.

Definition 9 (Dual). *For all kindable p and kinding environments Γ, we define the TST $\mathrm{co}_\Gamma(p)$ (in short, $\mathrm{co}(p)$ when $\Gamma = \emptyset$) in Figure 3.*

The following theorem states the soundness of the kind system: is particular, if the clock valuation ν_0 belongs to the kind of p, then p admits a compliant.

Theorem 4 (Soundness). *If $\vdash p : \mathcal{K}$ and $\nu \in \mathcal{K}$, then $(p, \nu) \bowtie (\mathrm{co}(p), \nu)$.*

Example 8. Recall the TST $q_1 = \mathtt{?a}\{x \leq 2\}.\,\mathtt{?b}\{x \leq 1\}$ in Example 6. We have:

$$\mathrm{co}(q_1) \;=\; \mathtt{!a}\{x \leq 1\}.\,\mathtt{!b}\{x \leq 1\}$$

Since $\vdash q_1 : \mathcal{K} = [\![x \leq 1]\!]$ and $\nu_0 \in \mathcal{K}$, by Theorem 4 we have that $q_1 \bowtie \mathrm{co}(q_1)$, as anticipated in Example 6.

The following theorem states the kind system is also complete: in particular, if p admits a compliant, then the clock valuation ν_0 belongs to the kind of p.

Theorem 5 (Completeness). *If $\vdash p : \mathcal{K}$ and $\exists q, \eta.\ (p, \nu) \bowtie (q, \eta)$, then $\nu \in \mathcal{K}$.*

Compliance is not transitive, in general (see [5]); however, the following Theorem 6 states that transitivity holds when passing through duals.

Theorem 6. *If $p \bowtie p'$ and $\mathrm{co}(p') \bowtie q$, then $p \bowtie q$.*

We now show that the dual is maximal w.r.t. the subtyping relation, like the dual in the untimed setting. We start by defining the semantic subyting preorder, which is a sound and complete model of the Gay and Hole subtyping relation (in reverse order) for untimed session types [4]. Intuitively, p is subtype of q if every q' compliant with q is compliant with p, too.

Definition 10 (Semantic subtyping). *For all TSTs p, we define the set $p^{\bowtie}$ as $\{q \mid p \bowtie q\}$. Then, we define the relation $p \sqsubseteq q$ whenever $p^{\bowtie} \supseteq q^{\bowtie}$.*

The following theorem states that $\mathrm{co}(p)$ is the maximum (i.e., the most "precise") in the set of the compliants of p, if not empty.

Theorem 7. $q \bowtie p \implies q \sqsubseteq \mathrm{co}(p)$

The following theorem reduces the problem of deciding $p \sqsubseteq q$ to that of checking compliance between p and $\mathrm{co}(q)$. Since both compliance and the dual construction are decidable, this implies decidability of subtyping.

Theorem 8. *If q admits a compliant, then: $p \sqsubseteq q \iff p \bowtie \mathrm{co}(q)$.*

4 Runtime monitoring

In this section we study runtime monitoring based on TSTs. The setting is the following: two participants A and B want to interact according to two (compliant) TSTs p_{A} and p_{B}, respectively. This interaction happens through a server, which monitors all the messages exchanged between A and B, while keeping track of the passing of time. If a participant (say, A) sends a message not expected by her TST, then the monitor classifies A as *culpable* of a violation. There are other two circumstances where A is culpable: (i) p_{A} is an internal choice, but A loses time until all the branches become unfeasible, or (ii) p_{A} is an external choice, but A does not readily receive an incoming message sent by B.

Note that the semantics in Figure 1 cannot be directly exploited to define such a runtime monitor, for two reasons. First, the synchronisation rule is purely symmetric, while the monitor outlined above assumes an asymmetry between internal and external choices. Second, the semantics in Figure 1 does not have transitions (either messages or delays) which are not allowed by the TSTs: for instance, $(\mathtt{!a}\{t \leq 1\}, \nu)$ cannot take any transitions (neither $\mathtt{!a}$ nor δ) if $\nu(t) > 1$. In a runtime monitor we want to avoid such kind of situations, where no actions are possible, and the time is frozen. More specifically, our desideratum is that the runtime monitor acts as a *deterministic* automaton, which reads a *timed trace* (a sequence of actions and time delays) and it reaches a unique state γ, which can be inspected to find which of the two participants (if any) is culpable.

To reach this goal, we define the semantics of the runtime monitor on two levels. The first level, specified by the relation $\twoheadrightarrow$, deals with the case of honest participants; however, differently from the semantics in Section 2, here we decouple the action of sending from that of receiving. More precisely, if A has an internal choice and B has an external choice, then we postulate that A must move first, by doing one of the outputs in her choice, and then B must be ready to do the corresponding input. The second level, called *monitoring semantics* and specified by the relation $\twoheadrightarrow_M$, builds upon the first one. Each move accepted by the first level is also accepted by the monitor. Additionally, the monitoring semantics defines transitions for actions not accepted by the first level, for instance unexpected input/output actions, and improper time delays. In these cases, the monitoring semantics signals which of the two participants is culpable.

Definition 11 (Monitoring semantics of TSTs). Monitoring configurations $\gamma, \gamma', \ldots$ *are terms of the form $P \parallel Q$, P and Q are triples (p, c, ν), where*

$$(\mathtt{!a}\{g,R\}.\,p \oplus p',[],\nu) \parallel (q,[],\eta) \xrightarrow{\mathsf{A}:\mathtt{!a}}\!\!\!\!\rightarrow (p,[\mathtt{!a}],\nu[R]) \parallel (q,[],\eta) \quad \text{if } \nu \in [\![g]\!] \qquad \text{[M-}\oplus\text{]}$$

$$(p,[\mathtt{!a}],\nu) \parallel (\mathtt{?a}\{g,R\}.\,q + q',[],\eta) \xrightarrow{\mathsf{B}:\mathtt{?a}}\!\!\!\!\rightarrow (p,[],\nu) \parallel (q,[],\eta[R]) \quad \text{if } \nu \in [\![g]\!] \qquad \text{[M-+]}$$

$$\frac{\nu+\delta \in \mathtt{rdy}(p) \qquad \eta+\delta \in \mathtt{rdy}(q)}{(p,[],\nu) \parallel (q,[],\eta) \xrightarrow{\delta}\!\!\!\!\rightarrow (p,[],\nu+\delta) \parallel (q,[],\eta+\delta)} \qquad \text{[M-Del]}$$

$$\frac{(p,c,\nu) \parallel (q,d,\eta) \xrightarrow{\lambda}\!\!\!\!\rightarrow (p',c',\nu') \parallel (q',d',\eta')}{(p,c,\nu) \parallel (q,d,\eta) \xrightarrow{\lambda}\!\!\!\!\rightarrow_M (p',c',\nu') \parallel (q',d',\eta')} \qquad \text{[M-Ok]}$$

$$\frac{(p,c,\nu) \parallel (q,d,\eta) \not\xrightarrow{\mathsf{A}:\ell}\!\!\!\!\rightarrow}{(p,c,\nu) \parallel (q,d,\eta) \xrightarrow{\mathsf{A}:\ell}\!\!\!\!\rightarrow_M (\mathbf{0},c,\nu) \parallel (q,d,\eta)} \qquad \text{[M-FailA]}$$

$$\frac{(d=[] \wedge \nu+\delta \notin \mathtt{rdy}(p)) \;\vee\; d \neq []}{(p,c,\nu) \parallel (q,d,\eta) \xrightarrow{\delta}\!\!\!\!\rightarrow_M (\mathbf{0},c,\nu+\delta) \parallel (q,d,\eta+\delta)} \qquad \text{[M-FailD]}$$

Fig. 4. Monitoring semantics (symmetric rules omitted).

p is either a TST or $\mathbf{0}$, and c is a one-position buffer (either empty or containing an output label). The transition relations $\twoheadrightarrow$ and $\twoheadrightarrow_M$ over monitoring configurations, with labels $\lambda, \lambda', \ldots \in (\{\mathsf{A},\mathsf{B}\} \times \mathtt{L}) \cup \mathbb{R}_{\geq 0}$, is defined in Figure 4.

In the rules in Figure 4, we always assume that the leftmost TST is governed by A, while the rightmost one is governed by B. In rule [M-⊕], A has an internal choice, and she can fire one of her outputs `!a`, provided that its buffer is empty, and the guard g is satisfied. When this happens, the message `!a` is written to the buffer, and the clocks in R are reset. Then, B can read the buffer, by firing `?a` in an external choice through rule [M-+]; this requires that the buffer of B is empty, and the guard g of the branch `?a` is satisfied. Rule [M-Del] allows time to pass, provided that the delay δ is permitted for both participants, and both buffers are empty. The last three rules specify the runtime monitor. Rule [M-Ok] says that any move accepted by $\twoheadrightarrow$ is also accepted by the monitor. Rule [M-FailA] is used when participant A attempts to do an action not permitted by $\twoheadrightarrow$: this makes the monitor evolve to a configuration where A is culpable (denoted by the term $\mathbf{0}$). Rule [M-FailD] makes A culpable when time passes, in two cases: either A has an internal choice, but the guards are no longer satisfiable; or she has an external choice, and there is an incoming message.

When both participants behave honestly, i.e., they never take [M-Fail*] moves, the monitoring semantics preserves compliance (Theorem 9). The *monitoring compliance* relation $\bowtie_M$ is the straightforward adaptation of that in Definition 6, except that $\twoheadrightarrow$ transitions are used instead of $\rightarrow$ ones (see [5]).

Theorem 9. $\bowtie \;= \bowtie_M$.

The following lemma establishes that the monitoring semantics is deterministic: that is, if $\gamma \xrightarrow{\lambda}\!\!\!\!\rightarrow_M \gamma'$ and $\gamma \xrightarrow{\lambda}\!\!\!\!\rightarrow_M \gamma''$, then $\gamma' = \gamma''$. Determinism is a very desirable property indeed, because it ensures that the culpability of a participant at any given time is uniquely determined by the past actions. Furthermore, for all finite timed traces $\boldsymbol{\lambda}$ (i.e., sequences of actions $\mathsf{A} : \ell$ or time delays δ), there exists some configuration γ reachable from the initial one.

Lemma 2. *Let $\gamma_0 = (p, [], \nu_0) \parallel (q, [], \eta_0)$. If $p \bowtie q$, then $(\twoheadrightarrow_M, \gamma_0)$ is deterministic, and for all finite timed traces $\boldsymbol{\lambda}$ there exists (unique) γ such that $\gamma_0 \xrightarrow{\boldsymbol{\lambda}}\!\!\!\!\rightarrow_M \gamma$.*

The goal of the runtime monitor is to detect, at any state of the execution, which of the two participants is culpable (if any). Further, we want to identify who is in charge of the next move. This is formalised by the following definition.

Definition 12 (Duties & culpability). *Let $\gamma = (p, c, \nu) \parallel (q, d, \eta)$. We say that A is* culpable *in γ iff $p = \mathbf{0}$. We say that A is* on duty *in γ if (i) A is not culpable in γ, and (ii) either p is an internal choice, or d is not empty.*

Lemma 3 guarantees that, in each reachable configuration, only one of the participants can be on duty; and if no one is on duty nor culpable, then both participants have reached success.

Lemma 3. *If $p \bowtie q$ and $(p, [], \nu_0) \parallel (q, [], \eta_0) \twoheadrightarrow_M^* \gamma$, then:*

1. *there exists at most one participant on duty in γ,*
2. *if there exists some culpable participants in γ, then no one is on duty in γ,*
3. *if no one is on duty in γ, then γ is success, or someone is culpable in γ.*

Note that both participants may be culpable in a configuration. E.g., let $\gamma = (\mathtt{!a}\{\mathtt{true}\}, [], \eta_0) \parallel (\mathtt{?a}\{\mathtt{true}\}, [], \eta_0)$. By applying [M-FailA] twice, we obtain:

$$\gamma \xrightarrow{\mathsf{A}:\mathtt{?b}}\!\!\!\!\rightarrow_M (\mathbf{0}, [], \nu_0) \parallel (\mathtt{?a}\{\mathtt{true}\}, [], \eta_0) \xrightarrow{\mathsf{B}:\mathtt{?b}}\!\!\!\!\rightarrow_M (\mathbf{0}, [], \nu_0) \parallel (\mathbf{0}, [], \eta_0)$$

and in the final configuration both participants are culpable.

Example 9. Let $p = \mathtt{!a}\{2 < t < 4\}$ be the TST of participant A, and let $q = \mathtt{?a}\{2 < t < 5\} + \mathtt{?b}\{2 < t < 5\}$ be that of B. We have that $p \bowtie q$. Let $\gamma_0 = (p, [], \nu_0) \parallel (q, [], \nu_0)$. A correct interaction is given by the timed trace $\eta = \langle 1.2,\ \mathsf{A} : \mathtt{!a},\ \mathsf{B} : \mathtt{?a} \rangle$. Indeed, $\gamma_0 \xrightarrow{\eta}\!\!\!\!\rightarrow_M (\mathbf{1}, [], \nu_0) \parallel (\mathbf{1}, [], \nu_0)$. On the contrary, things may go awry in three cases:

(i) a participant does something not permitted. E.g., if A fires $\mathtt{a}$ at 1 t.u., by [M-FailA]: $\gamma_0 \xrightarrow{1}\!\!\!\!\rightarrow_M \xrightarrow{\mathsf{A}:\mathtt{!a}}\!\!\!\!\rightarrow_M (\mathbf{0}, [], \nu_0 + 1) \parallel (q, [], \eta_0 + 1)$, where A is culpable.

(ii) a participant avoids to do something she is supposed to do. E.g., assume that after 6 t.u., A has not yet fired $\mathtt{a}$. By rule [M-FailD], we obtain $\gamma_0 \xrightarrow{6}\!\!\!\!\rightarrow_M (\mathbf{0}, [], \nu_0 + 6) \parallel (q, [], \eta_0 + 6)$, where A is culpable.

(iii) a participant does not receive a message as soon as it is sent. For instance, after $\mathtt{a}$ is sent at 1.2 t.u., at 5.2 t.u. B has not yet fired $\mathtt{?a}$. By [M-FailD], $\gamma_0 \xrightarrow{1.2}\!\!\!\!\rightarrow_M \xrightarrow{\mathsf{A}:\mathtt{!a}}\!\!\!\!\rightarrow_M \xrightarrow{4}\!\!\!\!\rightarrow_M (\mathbf{1}, [\mathtt{!a}], \nu_0 + 5.2) \parallel (\mathbf{0}, [], \eta_0 + 5.2)$, where B is culpable.

5 Conclusions

We have studied a theory of session types (TSTs), featuring timed synchronous communication between two endpoints. We have defined a decidable notion of compliance between TSTs, a decidable procedure to detect when a TST admits a compliant, a decidable subtyping relation, and a (decidable) runtime monitoring.

All these notions have been exploited in the design and development of a message-oriented middleware which uses TSTs to drive safe interactions among distributed components. The idea is a contract-oriented, bottom-up composition, where only those services with compliant contracts can interact via (binary) sessions. The middleware makes available a global store where services can advertise contracts, in the form of TSTs. Assume that A advertises a contract p to the store (this is only possible if p admits a compliant). A session between A and B can be established if (i) B advertises a contract q compliant with p, or (ii) B *accepts* the contract p (in this case, the contract of B is the dual of p). When the session is established, A and B can interact by sending/receiving messages through the session. During the interaction, all their actions are monitored (according to Definition 11), and possible misbehaviours are detected (according to Definition 12). The middleware is accessible through a set of public APIs; a suite of tools for developing contract-oriented applications is available at [5].

Related work. Compliance between TSTs is loosely related to the notion of compliance between *untimed* session types (in symbols, $\bowtie_u$). Let $u(p)$ be the session type obtained by erasing from p all the timing annotations. It is easy to check that the semantics of $(u(p), \nu_0) \mid (u(q), \nu_0)$ in Section 2 coincides with the semantics of $u(p) \mid u(q)$ in [4]. Therefore, if $u(p) \bowtie u(q)$, then $u(p) \bowtie_u u(q)$. Instead, *semantic conservation* of compliance does not hold, i.e. it is not true in general that if $p \bowtie q$, then $u(p) \bowtie_u u(q)$. E.g., let $p = \texttt{!a}\{t < 5\} \oplus \texttt{!b}\{t < 0\}$, and let $q = \texttt{?a}\{t < 7\}$. We have that $p \bowtie q$ (because the branch !b can never be chosen), whereas $u(p) = \texttt{!a} \oplus \texttt{!b} \not\bowtie_u \texttt{?a} = u(q)$. Note that, for every p, $u(\mathrm{co}(p)) = \mathrm{co}(u(p))$.

In the context of session types, time has been originally introduced in [9]. However, the setting is different than ours (multiparty and asynchronous, while ours is bi-party and synchronous), as well as its objectives: while we have focussed on primitives for the bottom-up approach to service composition [6], [9] extends to the timed case the *top-down* approach. There, a *choreography* (expressing the overall communication behaviour of a set of participants) is projected into a set of *session types*, which in turn are refined as processes, to be type-checked against their session type in order to make service composition preserve the properties enjoyed by the choreography.

Our approach is a conservative extension of untimed session types, in the sense that a participant which performs an output action chooses not only the branch, but the time of writing too; dually, when performing an input, one has to passively follow the choice of the other participant. Instead, in [9] external choices can also delay the reading time. The notion of correct interaction studied in [9] is called *feasibility*: a choreography is feasible iff all its reducts can reach the

success state. This property implies progress, but it is undecidable in general, as shown by [21] in the context of communicating timed automata (however, feasibility is decidable for the subclass of *infinitely satisfiable* choreographies). The problem of deciding if, given a local type T, there exists a choreography G such that T is in the projection of G and G enjoys (global) progress is not being addressed in [9]. We think that it can be solved by adapting our kind system (in particular rule [T-+] must be adjusted).

Another problem not addresses by [9] is that of determinining if a set of session types enjoys progress (which, as feasibility of choreographies, would be undecidable). In our work we have considered this problem, under a synchronous semantics, and with the restriction of two participants. Extending our semantics to an asynchronous one would make compliance undecidable (as it is for untimed asynchronous session types [15]). Note that our progress-based notion of compliance does not imply progress with the semantics of [9] (adapted to the binary case). For instance, let $p = \mathtt{?a}\{x \leq 2\}.\,\mathtt{!a}\{x \leq 1\}$ and $q = \mathtt{!a}\{y \leq 1\}.\,\mathtt{?a}\{y \leq 1\}$. We have that $p \bowtie q$, while in the semantics of [9] $(\nu_0, (p, q, \boldsymbol{w_0})) \rightarrow^* (\nu, (\mathtt{!a}\{x \leq 1\}, \mathtt{?a}\{y \leq 1\}, \boldsymbol{w_0}))$ with $\nu(x) = \nu(y) > 1$, which is a deadlock state.

Dynamic verification of timed multiparty session types is addressed by [24], where the top-down approach to service composition is pursued [20]. Our middleware instead composes and monitors services in a bottom-up fashion [6].

In [13] timed specifications are studied in the setting of *timed I/O transition systems* (TIOTS). They feature a notion of correct composition, called *compatibility*, following the *optimistic approach* pursued in [14]: roughly, two systems are compatible whenever there exists an environment which, composed with them, makes "undesirable" states unreachable. A notion of *refinement* is coinductively formalised as an alternating timed simulation. Refinement is a preorder, and it is included in the semantic subtyping relation (using compatibility instead of $\bowtie$). Because of the different assumptions (open systems and broadcast communications in [13], closed binary systems in TSTs), compatibility/refinement seem unrelated to our notions of compliance/subtyping. Despite the main notions in [13] are defined on semantic objects (TIOTS), they can be decided on timed I/O automata, which are finite representations of TIOTS. With respect to TSTs, timed I/O automata are more liberal: e.g., they allow for *mixed choices*, while in TSTs each state is either an input or an output. However, this increased expressiveness does not seem appropriate for our purposes: first, it makes the concept of culpability unclear (and it breaks one of the main properties of ours, i.e. that at most one participant is on duty at each execution step); second, it seems to invalidate any dual construction. This is particularly unwelcome, since this construction is one of the crucial primitives of contract-oriented interactions.

Acknowledgments. Work partially supported by Aut. Reg. of Sardinia L.R.7/2007 CRP-17285 (TRICS), P.I.A. 2010 ("Social Glue"), P.O.R. Sardegna F.S.E. Operational Programme of the Aut. Reg. of Sardinia, EU Social Fund 2007-13 – Axis IV Human Resources, Objective l.3, Line of Activity l.3.1), by MIUR PRIN 2010-11 project "Security Horizons", and by EU COST Action IC1201 "Behavioural Types for Reliable Large-Scale Software Systems" (BETTY).

References

1. PayPal buyer protection. `https://www.paypal.com/us/webapps/mpp/ua/useragreement-full#13`. Accessed: January 20, 2015.
2. R. Alur and D. L. Dill. A theory of timed automata. *Theor. Comput. Sci.*, 126(2):183–235, 1994.
3. F. Barbanera and U. de'Liguoro. Two notions of sub-behaviour for session-based client/server systems. In *PPDP*, pages 155–164, 2010.
4. F. Barbanera and U. de'Liguoro. Sub-behaviour relations for session-based client/server systems. *Math. Struct. in Comp. Science*, pages 1–43, 1 2015.
5. M. Bartoletti, T. Cimoli, M. Murgia, A. S. Podda, and L. Pompianu. Compliance and subtyping in timed session types. Technical report, 2015. `co2.unica.it`.
6. M. Bartoletti, E. Tuosto, and R. Zunino. Contract-oriented computing in CO_2. *Sci. Ann. Comp. Sci.*, 22(1), 2012.
7. G. Behrmann, A. David, and K. G. Larsen. A tutorial on Uppaal. In *Formal methods for the design of real-time systems*, pages 200–236. Springer, 2004.
8. J. Bengtsson and W. Yi. Timed automata: Semantics, algorithms and tools. In *ACPN*, pages 87–124, 2003.
9. L. Bocchi, W. Yang, and N. Yoshida. Timed multiparty session types. In *CONCUR*, pages 419–434, 2014.
10. G. Castagna, M. Dezani-Ciancaglini, E. Giachino, and L. Padovani. Foundations of session types. In *PPDP*, pages 219–230, 2009.
11. G. Castagna, N. Gesbert, and L. Padovani. A theory of contracts for web services. *ACM Transactions on Programming Languages and Systems*, 31(5), 2009.
12. R. Corin, P.-M. Deniélou, C. Fournet, K. Bhargavan, and J. J. Leifer. A secure compiler for session abstractions. *Journal of Computer Security*, 16(5), 2008.
13. A. David, K. G. Larsen, A. Legay, U. Nyman, L. Traonouez, and A. Wasowski. Real-time specifications. *STTT*, 17(1):17–45, 2015.
14. L. de Alfaro and T. A. Henzinger. Interface automata. In *ACM SIGSOFT*, pages 109–120, 2001.
15. P.-M. Deniélou and N. Yoshida. Multiparty compatibility in communicating automata: Characterisation and synthesis of global session types. In *ICALP*, 2013.
16. M. Dezani-Ciancaglini and U. de'Liguoro. Sessions and session types: An overview. In *WS-FM*, pages 1–28, 2009.
17. J. S. Dong, P. Hao, S. Qin, J. Sun, and W. Yi. Timed automata patterns. *IEEE Trans. Software Eng.*, 34(6):844–859, 2008.
18. T. A. Henzinger, X. Nicollin, J. Sifakis, and S. Yovine. Symbolic model checking for real-time systems. *Inf. Comput.*, 111(2):193–244, 1994.
19. K. Honda, V. T. Vasconcelos, and M. Kubo. Language primitives and type discipline for structured communication-based programming. In *ESOP*, 1998.
20. K. Honda, N. Yoshida, and M. Carbone. Multiparty asynchronous session types. In *POPL*, 2008.
21. P. Krcál and W. Yi. Communicating timed automata: The more synchronous, the more difficult to verify. In *CAV*, pages 249–262, 2006.
22. C. Laneve and L. Padovani. The *must* preorder revisited. In *CONCUR*, pages 212–225, 2007.
23. R. Milner. *Communication and concurrency*. Prentice Hall, 1989.
24. R. Neykova, L. Bocchi, and N. Yoshida. Timed runtime monitoring for multiparty conversations. In *BEAT*, pages 19–26, 2014.

25. K. Takeuchi, K. Honda, and M. Kubo. An interaction-based language and its typing system. In *PARLE*, 1994.
26. N. Yoshida, R. Hu, R. Neykova, and N. Ng. The Scribble protocol language. In *TGC*, pages 22–41, 2013.

A Modelling Paypal User Protection with TSTs

Following the lines of PayPal User Agreement [1], we now present the protection policy for buyers of an imaginary on-line payment platform. The reason why we do not use PayPal User Agreement itself is because it lacks of some details which are mandatory when formally describing a process. First, Paypal User Agreement says that if PayPal finds in your favor on your claim, you will be refunded, but it does not specify which are the points to have the claim in one's favor. Second, it does not specify a deadline for the refund to be sent. Finally, there is some ambiguity: the policy protects a customer if the item does not arrive, or if it is different from what expected, but it does not specify what to do in case a buyer opens a dispute for an item not received which eventually arrives and it is not what expected. All those missing details are easily managed during human interaction; but are a hindrance when dealing with automatic processes.

So, back to our example, assume our imaginary on-line payment platform is called PayNow. PayNow helps customers in on-line purchasing, providing protection against misbehaviours. In case a buyer has not received what they have paid for, or if they have received something significantly different from what they wanted, they can ask PayNow to solve the issue by opening a dispute. The dispute can be opened within 180 days of the date the buyer made the payment. After opening the dispute, the buyer and the seller may try to come to an agreement, or the item will finally arrive. If an agreement is not found within 20 days, the buyer can escalate the dispute to a claim. However, in case of an item not received, the buyer must wait at least 7 days from the date of payment to escalate a dispute. Upon not reaching an agreement, if still the buyer does not escalate the dispute to a claim within 20 days, the dispute is considered aborted.

In case of an item not received, PayNow will ask the buyer to provide documentation to support the payment made, within 3 days of the date the dispute was escalated to a claim. After that, the payment will be refunded, within a week. In case of an item "significantly not as described", PayNow will ask the buyer to provide documentation to support the claim, (as for instance, a photo of the item), within 3 days of the date the dispute was ecaladed to a claim. Upon evaluating the documentation, PayNow will ask the buyer either to destroy the item and to provide proof of destruction, or to ship the item back to the seller and to provide proof of delivery. These operations must be accomplished within 3 days of the date the request has been made. If the buyer complies with providing documentation in a timely manner, PayNow will refund them, within a week. It may happen that a buyer open a dispute for an item not arrived, and in the meanwhile the item arrives but it is not as described. In this case, the dispute can be escalated to a claim for a "significantly not as described" item.

Here are the actions of the contract:

`pay`: the buyer has paid for the item;
`ok`: the buyer and the seller have come to an agreement;
`inr`: a dispute has been opened because of "Item Not Received";
`snad`: a dispute has been opened because of the item was "Significantly Not as Described";

`claimINR`: the buyer escalates the dispute to a claim;
`claimSNAD`: the buyer escalates the dispute to a claim;
`rcpt`: the buyer provides proof of the payment;
`refund`: the buyer is refunded;
`abort`: the dispute process is aborted;
`photo`: the buyer provides photo of the item;
`sendBack`: the buyer is asked to send the item back;
`ackSendBack`: the buyer provides proof of delivery;
`destroy`: the buyer is asked to destroy the item;
`ackDestroy`: the buyer provides proof of destruction.

The following clocks are used:

t_{pay}: initialized at the moment in which the buyer has paid for the item
t_{inr}: initialized at the moment the dispute is opened for "item not received"
t_{snad}: initialized at the moment the dispute is opened for "item significantly not as described"
t_c: used to time actions during the claim procedure

The contract of PayNow is described by the following TST:

$$
\begin{aligned}
&\texttt{?pay}\{\texttt{true}, t_{pay}\}.(\\
&\quad \texttt{?ok}\\
&\quad + \texttt{?inr}\{t_{pay} < 180, t_{inr}\}.(\texttt{?ok}\{t_{inr} < 20\}\\
&\qquad + \texttt{?claimINR}\{t_{inr} < 20 \wedge t_{pay} > 7, t_c\}.\\
&\qquad\quad \texttt{?rcpt}\{t_c < 3, t_c\}.\texttt{!refund}\{t_c < 7\}\\
&\qquad + \texttt{?claimSNAD}\{t_{inr} < 20, t_c\}.\texttt{?photo}\{t_c < 7\}.(\\
&\qquad\quad \texttt{!sendBack}\{\texttt{true}, t_c\}.\texttt{?ackSendBack}\{t_c < 3\}.\texttt{!refund}\\
&\qquad\quad \oplus\ \texttt{!destroy}\{\texttt{true}, t_c\}.\texttt{?ackDestroy}\{t_c < 3\}.\texttt{!refund})\\
&\qquad + \texttt{?abort})\\
&\quad + \texttt{?snad}\{t_{pay} < 180, t_{snad}\}.(\texttt{?ok}\{t_{snad} < 20\}\\
&\qquad + \texttt{?claimSNAD}\{t_{snad} < 20, t_c\}.\texttt{?photo}\{t_c < 7\}.(\\
&\qquad\quad \texttt{!sendBack}\{\texttt{true}, t_c\}.\texttt{?ackSendBack}\{t_c < 3\}.\texttt{!refund}\\
&\qquad\quad \oplus\ \texttt{!destroy}\{\texttt{true}, t_c\}.\texttt{?ackDestroy}\{t_c < 3\}.\texttt{!refund})\\
&\qquad + \texttt{?abort})
\end{aligned}
$$

Let us consider a possible contract for buyer Alice. Alice is willing to wait 10 days to receive the item she has paid for, but after that she will open a claim. Alice will instantly provide PayNow with every documentation they may need in order to issue the refund. Also in case Alice receives an item significally different from what she has paid for, she will complain to PayNow by opening a claim, as soon as the item is received. Alice will timely comply to do whatever PayNow requires (either to destroy the item or to send it back) in order to be refunded. Alice's contract is described by the following TST:

$$
\begin{aligned}
&\texttt{!pay}\{\texttt{true}, t_{pay}\}.\\
&\quad (\ \texttt{!ok}\{t_{pay} < 10\}\\
&\quad \oplus \texttt{!inr}\{t_{pay} = 10, t_{inr}\}.\texttt{!claimINR}\{t_{inr} = 10\}.\texttt{!rcpt}\{t_{inr} = 10, t_c\}.\texttt{?refund}\\
&\quad \oplus \texttt{!snad}\{t_{pay} < 10, t_{snad}\}.\texttt{!claimSNAD}\{t_{snad} = 0\}.\texttt{!photo}\{t_{snad} = 0\}.\\
&\qquad (\ \texttt{?sendback}\{t_c\}.\texttt{!acksendback}\{t_c < 3\}.\texttt{?refund}\\
&\qquad + \texttt{?destroy}\{t_c\}.\texttt{!ackdestroy}\{t_c < 3\}.\texttt{?refund}\ )
\end{aligned}
$$

Alice's and PayNow's contracts are compliant.

Let us consider another buyer: Bob. Bob is willing to wait 10 days to receive the item he has paid for, but after that, he will open a dispute. If he and the seller are able to came to an agreement within 4 days, or if in the meanwhile the item arrives, he will close the dispute. Otherwise, he will escalate the dispute to a claim. In case Bob receive an item significally different from what he has paid for, he will complain to PayNow opening a dispute. Also in this case, he will wait 4 days for the seller to solve the problem, after which he will escalate the dispute to a claim. Bob will do whatever he is asked to, as soon as he is asked.

$$
\begin{array}{l}
\texttt{!pay}\{\texttt{true}, t_{pay}\}. \\
\quad (\ \texttt{!ok}\{t_{pay} < 10\} \\
\quad \oplus \texttt{!inr}\{t_{pay} = 10, t_{inr}\}. \\
\qquad (\ \texttt{!ok}\{t_{inr} < 14\} \\
\qquad \oplus \texttt{!claimINR}\{t_{inr} = 14, t_c\}.\texttt{!rcpt}\{t_c = 14, t_c\}.\texttt{?refund}\ ) \\
\qquad \oplus \texttt{!claimSNAD}\{t_{snad} < 14, t_c\}.\texttt{!photo}\{t_c < 14\}. \\
\qquad\quad (\ \texttt{?sendback}\{t_c\}.\texttt{!acksendback}\{t_c < 7\}.\texttt{?refund} \\
\qquad\quad + \texttt{?destroy}\{t_c\}.\texttt{!ackdestroy}\{t_c < 7\}.\texttt{?refund}\ ) \\
\quad + \texttt{!snad}\{t_{pay} < 10, t_{snad}\}. \\
\qquad (\ \texttt{!ok}\{t_{snad} < 4\} \\
\qquad \oplus \texttt{!claimSNAD}\{t_{snad} \geq 4, t_c\}.\texttt{!photo}\{t_c < 7\}. \\
\qquad\quad (\ \texttt{?sendback}\{t_c\}.\texttt{!acksendback}\{t_c < 1\}.\texttt{?refund} \\
\qquad\quad + \texttt{?destroy}\{t_c\}.\texttt{!ackdestroy}\{t_c < 1\}.\texttt{?refund}\ )
\end{array}
$$

Also Bob's and PayNow's TSTs are compliant.

B Proofs for Section 2

In this section, we proceed to prove that compliance between TSTs is decidable. In order to do so, we map TSTs into Timed Automata and we prove that the resulting network is deadlock-free if and only if the TSTs are compliant (Theorem 11). The mapping has been actually implemented using Uppaal Timed Automata [7].

To ease the process of mapping a TST into an automata, we express recursion in TSTs through defining equations([23]), which are equivalent to the representation with the rec construct.

Here is the roadmap of this section:

- In Appendix B.1, we present a variant of TSTs which uses defining equations instead of the rec construct. Although the two variants are equivalent, the one using defining equations is best suited to the purpose of translating TSTs into **TA**s. We also introduce a new semantics $\rightarrow_D$ (Figure 5), which preserve compliance (Lemma 6).
- In Appendix B.2, we define the semantics of a networks of **TA**s.
- In Appendix B.3, we present some patterns used to define **TA**s, in both graphic and non-graphic way (Figure 7). These patterns will be useful to express the mapping in a compact and readable form.

– In Appendix B.4, we sum up things together and we define the mapping of two TSTs, in Definition 28.
– In Appendix B.5, we prove that $\rightarrow_D$ is bisimilar to the automata network obtained by the mapping. Hence, we finally prove, in Theorem 11, that the resulting automata network is deadlock free if and only if the original TSTs are compliant (Theorem 11).
– In Appendix B.6 we actually implement the mapping using UPPAAL [7] and we discuss how to check compliance.

B.1 Defining equations

In this section, we present an alternative treatment of recursion, which uses *defining equations* ([23]) to express infinite processes. This definitional mechanism is a standard way to define recursion without the use of rec and it is more convenient for our purposes.

With defining equations, a TST process is represented through a couple: a recursion variable and a set of defining equations of the form $X_i \triangleq p_i$ where X_i is a recursion variable and p_i is a term of the syntax in Definition 13.

Definition 13 (Defining-equation normal form). *A process in defining-equation normal form (DENF) is as a couple (X, D) where D is a set of defining equations (DE), each of the form $X' \triangleq p$, where:*

$$p \quad ::= \quad \mathbf{1} \quad \mid \quad \bigoplus_{i \in I} \mathtt{!a_i}\{g_i, R_i\}.Y_i \quad \mid \quad \sum_{i \in I} \mathtt{?a_i}\{g_i, R_i\}.Y_i$$

and (i) the index set I is finite and non-empty, and (ii) the actions in internal/external choices are pairwise distinct

Given a DENF (X, D), we indicate with $\mathtt{urv}(D)$ the *used* recursion variable in D; and with $\mathtt{rv}(D)$ the defined recursion variables in D. For instance, let $D = \{X \triangleq \mathtt{!a}.Y\}$); then $\mathtt{urv}(D) = \{Y\}$ while $\mathtt{rv}(D) = \{X\}$.

Definition 14 (Well-formed). *We say that (X, D) is* well-formed*, if (i) $X \in \mathtt{rv}(D)$; (ii) if $\{X\} \cup \mathtt{urv}(D) = \mathtt{rv}(D)$ and (iii) if all the used recursion variables in D are defined exactly once.*

For instance $(X, \{X \triangleq \mathtt{!a}.X\})$ is well-formed, while $(X, \{Y \triangleq \mathbf{1}\})$, $(X, \{X \triangleq \mathtt{!a}.X, X \triangleq \mathtt{!b}.X\})$ and $(X, \{X \triangleq \mathtt{!a}.Y\})$ are not.

Every TST, can be expressed through a DENF. For instance, the DENF of $p = \mathrm{rec}\, X.\, \mathtt{?a}.\mathtt{?b}.X$ is $(X, \{X \triangleq \mathtt{?a}.Y, Y \triangleq \mathtt{?b}.X\})$, where every recursion variable is guarded by exactly one action. The algorithm to obtain the DENF of a TST is presented in Definition 15.

Definition 15 (DENF transformation). *Let p be a TST process. Let V be an infinite set of recursion variables fresh in p. Then, the DENF of p, namely*

$\langle p \rangle_V$, *is given by :*

$$
\begin{aligned}
\langle \bigcirc \alpha\{g_i, R_i\}.p_i \rangle_V = \; & (X_0,\ \textstyle\bigcup_{i=1}^{n} D_i \cup \{X_0 \triangleq \bigcirc_{i=1}^{n} \alpha_i\{g_i, R_i\}.X_i\}) \\
& \quad \textit{for } \bigcirc \in \{\oplus, \textstyle\sum\} \textit{ with } X_0 \in V \textit{ and} \\
& \quad \langle p_1 \rangle_{V \setminus \{X_0\}} = (X_1, D_1) \\
& \quad \langle p_i \rangle_{V \setminus W} = (X_i, D_i) \\
& \quad \textit{where } W = (\{X_0\} \cup \textstyle\bigcup_{j=1}^{i-1} \mathtt{rv}(D_j)) \\
\langle \mathrm{rec}\, X\,.\, p' \rangle_V \quad & = (X', D[X'/X_0, X'/X]) \quad \textit{where } X' \in V \textit{ and} \\
& \quad \langle p' \rangle_{V \setminus X'} = (X_0, D) \\
\langle X \rangle_V \quad & = (X, \emptyset) \\
\langle \mathbf{1} \rangle_V \quad & = (X_0, \{X_0 = \mathbf{1}\}) \quad \textit{and } X_0 \in V
\end{aligned}
$$

For short, we indicate the normal form of p, with $\langle p \rangle$.

Example 10. Here we present some examples of TSTs and their DENF: we omit guards and reset sets to focus on the transformation.

(1) $p_1 = \mathtt{!a} \oplus \mathtt{!b}$
$\langle p_1 \rangle = (X_0, \{X_0 \triangleq \mathtt{!a}.X_1 \oplus \mathtt{!b}.X_2,\ X_1 \triangleq \mathbf{1},\ X_2 \triangleq \mathbf{1}\})$

(2) $p_2 = \mathtt{!a}.\mathrm{rec}\, X\,.\, (\mathtt{!b}.X \oplus \mathtt{!c})$
$\langle p_2 \rangle = (X_0, \{X_0 \triangleq \mathtt{!a}.X_1,\ X_1 \triangleq \mathtt{!b}.X_1 \oplus \mathtt{!c}.X_2,\ X_2 \triangleq \mathbf{1}\})$

(3) $p_3 = \mathrm{rec}\, X\,.\, \mathtt{!a}.\mathrm{rec}\, Y\,.\, (\mathtt{!b}.X \oplus \mathtt{!c}.Y)$
$\langle p_3 \rangle = (X_0, \{X_0 \triangleq \mathtt{!a}.X_1,\ X_1 \triangleq \mathtt{!b}.X_0 \oplus \mathtt{!c}.X_1\})$

(4) $p_4 = \mathrm{rec}\, X\,.\, (\mathtt{!a}.\mathrm{rec}\, X\,.\, \mathtt{!b}.X \ \oplus \ \mathtt{!c}.X)$
$\langle p_4 \rangle = (X_0, \{X_0 \triangleq \mathtt{!a}.X_1 \oplus \mathtt{!c}.X_0,\ X_1 \triangleq \mathtt{!b}.X_1\})$

Without loss of generality, we assume that TSTs have not multiple unguarded rec definitions, (like for instance $\mathrm{rec}\, X.\, \mathrm{rec}\, Y.\, p$), given that they can be collapsed into one single definition without loss of expressiveness (see Lemma 4).

Lemma 4. $\mathrm{rec}\, X.\, \mathrm{rec}\, Y.\, p \equiv \mathrm{rec}\, X.\, p[X/Y]$

If a TST has no free-variables, its transformation will generate a well-formed DENF.

Definition 16 (Free variables). *Let p be a TST process. We indicate with $\mathtt{fv}(p)$ the free variables in p. Formally:*

$$
\mathtt{fv}(p) = \begin{cases}
\bigcup_i \mathtt{fv}(p_i) & \textit{if } p = \bigcirc \alpha\{g, R\}.p_i \textit{ with } \bigcirc \in \{\oplus, \textstyle\sum\} \\
\mathtt{fv}(p') \setminus X & \textit{if } p = \mathrm{rec}\, X\,.\, p' \\
\{X\} & \textit{if } p = X \\
\emptyset & \textit{if } p = \mathbf{1}
\end{cases}
$$

We say that a TST process p is closed *if it contains no free variables.*

Lemma 5 (DENF well-formed transformation). *If p is a closed TST, then $\langle p \rangle$ is a well-formed DENF.*

In the following of the section, we will assume that DENFs are well-formed.

New semantics for DENF. We now introduce a new semantics for DENF, which preserve compliance.

Definition 17. *The set $\mathcal{S}$ of state terms, ranged over by $x, y \ldots$, is defined as follow:*

$$x \quad ::= \quad \mid \tau X \mid [\mathtt{!a}\{g, R\}]\, X \mid X$$

where $\mathtt{a} \in \mathtt{A}$, g is a guard and R is a reset set.

Definition 18 (DENF semantics). *Let (X, D') and (Y, D'') be two DENFs with $\mathtt{rv}(D') \cap \mathtt{rv}(D'') = \emptyset$. Let $s, r \in \mathcal{S}$. Let $D = D' \cup D''$. Then the relation $\rightarrow_D$ is inductively defined by the set of rules in Figure 5.*

We now comment the rules in Figure 5. The first six rules are auxiliary, as they describe the behaviour of a DENF in isolation. While the last four rules deal with configurations of two DENFs. Aside from obvious syntactic differences caused by DENF, $\rightarrow_D$ is very similar to $\rightarrow$. Rule [DE-$\oplus$] allows a TST to commit to the branch $\mathtt{!a}$ of her internal choice, provided that the corresponding guard is satisfied in the clock valuation ν. This results in the term $[\mathtt{!a}\{g, R\}]\, p$, which can only fire $\mathtt{!a}$ through rule [DE-!]; when this happens, the clocks in R are reset. Rule [DE-?] allows an external choice to fire any of its input actions whose guard is satisfied. Rule [DE-DEL1] and [DE-DEL2] allows time to pass; this is always possible for external choices and success term, while for an internal choice we require that at least one of the guards remains satisfiable; this is obtained through the function $\mathtt{rdy}$ in Definition 5. Time cannot pass for committed choices. While all these rules have their corresponding in $\rightarrow$; rule [DE-*empty*] has not: it obliges every internal choice to perform an empty step before committing to a branch. So, prior to every commit step [DE-$\oplus$], there must be exactly one step [DE-*empty*].

Rule [DES-$\oplus$] allows a DENF to perform an empty step or to commit in an internal choice. Rule [DES-DEL] allows time to pass, equally for both endpoints. Rule [DES-τ] is the standard synchronisation rule with the difference that the label is not τ but it shows the action performed. Finally, rule [DES-$\checkmark$] has not corresponding in $\rightarrow$: it allows for a never-ending synchronization in case both DENF reach the $\mathbf{1}$ term.

All the differences have been introduced to allow a strong bisimilarity with the automata network of section Appendix B.4, but they do not affect compliance.

Notation 1 *As usual, we overload the symbol $\rightarrow$ to denote both the LTS and its transition relation. To make explicit that some state q belongs to the LTS $\rightarrow$, we write the state as a pair $(q, \rightarrow)$.*

Definition 19 (Compliance). *Let D be a set of DEs. We say that $((x, \nu) \mid (y, \eta), \rightarrow_D)$ is* deadlock *whenever there is no δ and no $\alpha \in \mathtt{A} \cup \{\tau\}$ such that: $(x, \nu + \delta) \mid (y, \eta + \delta) \xrightarrow{\alpha}_D$.*

$$\frac{(X \triangleq p) \in D \quad p = \oplus \ldots \quad \nu \in \mathtt{rdy}(p)}{(X,\ \nu) \xrightarrow{\tau}_D (\tau X,\ \nu)} \quad [\text{DE-}\mathit{empty}]$$

$$\frac{(X \triangleq \mathtt{!a}\{g, R\}.\, Y \oplus p) \in D \qquad \nu \in [\![g]\!]}{(\tau X,\ \nu) \xrightarrow{\tau}_D ([\mathtt{!a}\{g, R\}]\, Y,\ \nu)} \quad [\text{DE-}\oplus]$$

$$([\mathtt{!a}\{g, R\}]\, Y,\ \nu) \xrightarrow{\mathtt{!a}}_D (Y,\ \nu[R]) \quad [\text{DE-!}]$$

$$\frac{(X \triangleq \mathtt{?a}\{g, R\}.\, Y + p) \in D \qquad \nu \in [\![g]\!]}{(X,\ \nu) \xrightarrow{\mathtt{?a}}_D (Y,\ \nu[R])} \quad [\text{DE-?}]$$

$$\frac{(X \triangleq + \ldots) \in D \ \vee\ (X \triangleq \mathbf{1}) \in D}{(X,\ \nu) \xrightarrow{\delta}_D (X, \nu + \delta)} \quad [\text{DE-DEL1}]$$

$$\frac{X \triangleq p \in D \quad \nu + \delta \in \mathtt{rdy}(p)}{(\tau X,\ \nu) \xrightarrow{\delta}_D (\tau X,\ \nu + \delta)} \quad [\text{DE-DEL2}]$$

$$\frac{(x, \nu) \xrightarrow{\tau}_D (x', \nu)}{(x, \nu) \mid (y, \eta) \xrightarrow{\tau}_D (x', \nu) \mid (y, \eta)} \quad [\text{DES-}\oplus]$$

$$\frac{(x, \nu) \xrightarrow{\delta}_D (x, \nu') \quad (y, \eta) \xrightarrow{\delta}_D (y, \eta')}{(x, \nu) \mid (y, \eta) \xrightarrow{\delta}_D (x, \nu') \mid (y, \eta')} \quad [\text{DES-DEL}]$$

$$\frac{(x, \nu) \xrightarrow{\mathtt{!a}}_D (x', \nu') \quad (y, \eta) \xrightarrow{\mathtt{?a}}_D (y', \eta')}{(x, \nu) \mid (y, \eta) \xrightarrow{\mathtt{a}}_D (x', \nu') \mid (y', \eta')} \quad [\text{DES-}\tau]$$

$$\frac{X \triangleq \mathbf{1} \in D \quad Y \triangleq \mathbf{1} \in D}{(X, \nu) \mid (Y, \eta) \xrightarrow{\checkmark}_D (X, \nu) \mid (Y, \eta)} \quad [\text{DES-}\checkmark]$$

Fig. 5. Semantics of timed session types with the use of defining equations (symmetric rules omitted).

We then write $(x, \nu) \bowtie_D (y, \eta)$ *whenever:*

$$(x, \nu) \mid (y, \eta) \rightarrow_D^* (x', \nu') \mid (y', \eta') \quad \textit{implies} \quad (x', \nu') \mid (y', \eta') \textit{ not deadlock}$$

We say that x *and* y *are* compliant *whenever* $(x, \nu_0) \bowtie_D (y, \eta_0)$ *(in short,* $x \bowtie_D y$*).*

Lemma 6. *Let* p *and* q *be two TST with no shared clocks and no free variables. Then:*

$$p \bowtie q \iff \langle p \rangle \bowtie_D \langle q \rangle$$

B.2 Timed automata

A timed automaton (**TA**) [2] is a finite state automaton extended with a set of clocks which progress all with the same pace and which can be reset. **TA** are equipped with a set of locations and a set of edges. Every automaton has an initial location. The current location is the location an automaton is in, during a computation. Every location is associated with a clock constraints called *invariant*. The system can stay in a location only if the invariant of that location is satisfied by the clock valuation. Invariants are used to ensure the progress of the model, that is, the system cannot stay in a location forever. Every edge is associated with a clock constraints called *guard*, a label, and a *reset set*: an edge can be taken only if the clock valuation satisfies the guard. Once the edge has been taken, the label is fired and all the clocks in the reset set are reset to zero.While taking an edge is an instantaneous action, time can elapse in a location.

A network is a set of **TA**s which can interact by synchronizing on channels. The *current state* of a network is given by the set of the current locations each automaton is in. Channels are associated to edges by the edge label. Two automata can synchronize on a *binary channel* `a` if (i) their current locations are entering edges with complementary labels (such as `!a` and `?a`) and if (ii) the guards of those edges are satisfied by the clock valuation. Edges with the empty label τ are not associated to any channel.

Notwithstanding with the wide variety of specifications for automata appeared in literature, we equip our automata with some non standard features. First of all, we assume locations can be *urgent*. If a location is urgent, time cannot pass when the system is in that location. As a consequence of this, binary channel starting from urgent locations are compelled to synchronize as soon as the synchronization is available. Secondly, in order to avoid over technicalities in proofs, we prevent disjunction in guards while we assume it in invariants. In Appendix B.6 we will discuss how to deal with these constraints in an actual implementation.

Formally, a **TA** is a tuple $A = (Loc, Loc^u, l_0, E, I)$ where Loc is a set of locations; $Loc^u \subseteq Loc$ is the set of urgent locations; $l_0 \in Loc$ is the initial location; $E = Loc \times L \times \mathcal{GC} \times \wp(\mathbb{C}) \times Loc$ is a set of edges; $L = \mathtt{A}^! \cup \mathtt{A}^? \cup \{\tau\}$; and $I : Loc \to \mathcal{GC}$ is the invariant function. Given an edge $e = (l, \alpha, g, R, l') \in E$, l is the source location, $\alpha \in L$ is the label; g is the guard; R is the reset set and l' is the target location.

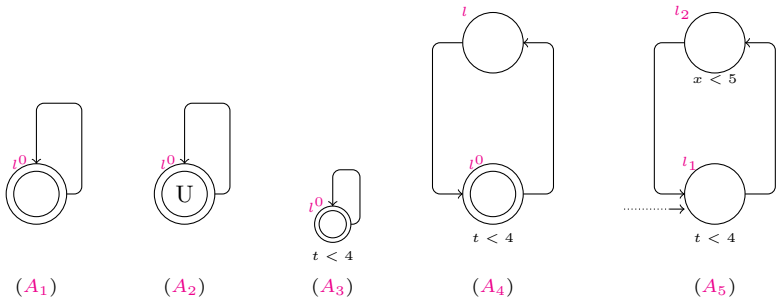

Fig. 6. Automata for Example 11 on deadlock-freeness.

Given a network N, the relation $\rightarrow_N$ is defined by the set of rules in Definition 20.

Definition 20. *Let* $A_i = (Loc_i, Loc_i^u, l_i^0, E_i, I_i)$ *for* $i \in \{1,2\}$ *be* **TA***s. The behaviour of the network composed by* A_1 *and* A_2 *(denoted with* $A_1 \mid A_2$*) is given by the timed transition system* $(S, s_0, Lab, \rightarrow_N)$*, where* $S = Loc_1 \times Loc_2 \times \mathbb{V}$*;* $s_0 = (l_1^0, l_2^0, \nu_0)$*;* $Lab = \mathtt{A} \cup \{\tau\} \cup \mathbb{R}_{\geq 0}$*, and* $\rightarrow_N$ *is defined as follows (symmetric rules omitted):*

$$\frac{(l_1, \tau, g, R, l'_1) \in E_1 \qquad \nu \in [\![g]\!] \qquad \nu[R] \in [\![I_1(l'_1) \wedge I_2(l_2)]\!]}{(l_1, l_2, \nu) \xrightarrow{\tau}_N (l'_1, l_2, \nu[R])} \quad \text{[TA1]}$$

$$\frac{\begin{array}{c}(l_1, \mathtt{!a}, g_1, R_1, l'_1) \in E_1 \\ (l_2, \mathtt{?a}, g_2, R_2, l'_2) \in E_2\end{array} \qquad \nu \in [\![g_1 \wedge g_2]\!] \qquad \nu[R_1][R_2] \in [\![I_1(l'_1) \wedge I_2(l'_2)]\!]}{(l_1, l_2, \nu) \xrightarrow{\mathtt{a}}_N (l'_1, l'_2, \nu[R_1][R_2])} \quad \text{[TA2]}$$

$$\frac{\forall i \in \{1,2\} : (\nu + \delta) \in [\![I_i(l_i)]\!] \wedge l_i \notin Loc_i^u}{(l_1, l_2, \nu) \xrightarrow{\delta}_N (l_1, l_2, \nu + \delta)} \quad \text{[TA3]}$$

Checking deadlock property. In a network of **TA**s, the current state is *deadlock* if there are no *action*-transitions possible; neither at the current moment, nor with the time elapsing.

Definition 21. *Let* $N = A_1 \mid A_2$ *be a network of* **TA***s. A state* $(s, \rightarrow_N)$ *is* deadlock *iff* $\nexists \delta \geq 0,\ \mathtt{a}_\tau \in \mathtt{A} \cup \{\tau\} :\quad s \xrightarrow{\delta}_N \xrightarrow{\mathtt{a}_\tau}_N s'$

A network of **TA**s, is *deadlock-free* if no one of the reachable states of the system is a deadlock state.

In the following example we consider the deadlock property of single automaton in isolation.

Example 11. In Figure 6, we have a graphical representation of five automata. Locations are depicted with a circle; a double circle represent an initial location;

the symbol U indicates a urgent location. If no invariant is specified, then it is considered to be *true*. An arrow from a location to another represents an edge: if no label is specified, then it is considered to be τ; if no guard is specified it is considered to hold *true*; if no reset set is specified it is considered to be empty.

1. Let $A_1 = (\{l^0\}, \emptyset, l^0, \{e_1\}, I)$ with $e_1 = \{(l^0, \texttt{true}, \tau, \emptyset, l^0)\}$. The edge e_1 is always enabled as the time passes, hence a τ-transition is always possible and the automata is *not* deadlock.
2. Let $A_2 = (\{l^0\}, \{l^0\}, l^0, \{e_1\}, I)$ with $e_1 = \{(l^0, \texttt{true}, \tau, \emptyset, l^0)\}$. Here, time is not allowed to pass, nevertheless the edge e_1 is always enabled. Hence a τ-transition is always possible and the automata is *not* deadlock.
3. Let $A_3 = (\{l^0\}, \emptyset, l^0, \{e_1\}, I)$ with $I(l^0) = (t < 4)$ and $e_1 = \{(l^0, \texttt{true}, \tau, \emptyset, l^0)\}$. In location l^0, time cannot pass past 4, nevertheless the edge e_1 is always enabled in that time frame. Hence a τ-transition is always possible and the automata is *not* deadlock.
4. Let $A_4 = (\{l^0, l\}, \emptyset, l^0, \{e_1, e_2\}, I)$ with $I(l^0) = (t < 4)$, $e_1 = \{(l^0, \texttt{true}, \tau, \emptyset, l)\}$ and $e_2 = \{(l, \texttt{true}, \tau, \emptyset, l^0)\}$. In location l^0, time cannot pass past 4, while in location l time has no upper bound. If the system enters location l and let time pass past 4, then a τ-transition is no more possible because of the invariant on l^0. Hence the automata *is* deadlock.
5. Let $A_5 = (Loc, \emptyset, l^0, E, I)$. In Figure 6 we see a portion of automata A_5. We focus on the two locations l_0 and l_1 with invariants $I(l_1) = (t < 4)$ and $I(l_2) = (x < 5)$ related to two different clocks. In location l_1, time cannot pass if clock t is past 4, while in location l_2 time cannot pass if clock x is past 5. If – due to possible previous resets operations – when entering location l_1 we have that the clock valuation ν of x and t differs of only by one unit i.e. $\nu(x) - \nu(t) = 1$ then the τ-transitions are always possible. Otherwise, the system will end in a deadlock state as in the previous (4) example.

B.3 TA Patterns

In the following we introduce some patterns which will be useful in Appendix B.4, when dealing with automata composition. Graphically, we will represent these patterns as in [17]: locations are depicted with a circle; a double circle represent an initial location; the symbol U indicates a urgent location. If no invariant is specified, then it is considered to be *true*. An arrow from a location to another represents an edge: if no label is specified, then it is considered to be τ; if no guard is specified it is considered to hold *true*; if no reset set is specified it is considered to be empty. An automaton is depicted as a triangle; the left vertex of a triangle represents its initial location; an arrow from a location to a triangle represents an edge pointing to the initial location of the automaton in the triangle.

The *Idle* pattern creates an automaton with only a location.

Definition 22 (Idle pattern). *Let $l^0 \in Loc$. Then, $Idle(l^0) = (\{l^0\}, \emptyset, l^0, \emptyset, \emptyset)$.*

The *Su* pattern creates an automaton with a location and two self loop synchronizing on the special channel $\checkmark$.

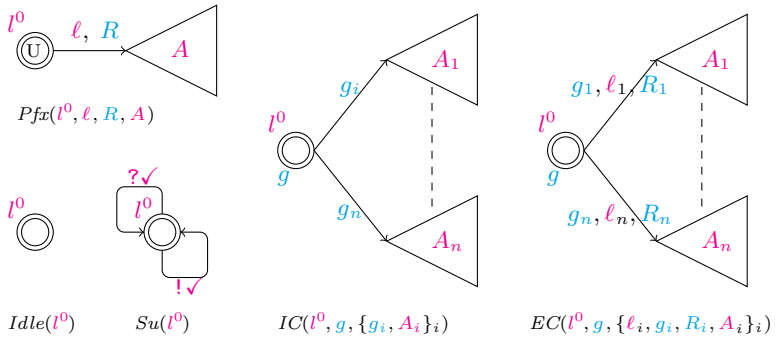

Fig. 7. Automata patterns.

Definition 23 (Success pattern). *Let* $l^0 \in Loc$. *Then,*

$$Su(l^0) = (\{l^0\}, \emptyset, l^0, E, \emptyset)$$

where $E = \{(l^0, \texttt{!}\checkmark, \texttt{true}, \emptyset, l^0), (l^0, \texttt{?}\checkmark, \texttt{true}, \emptyset, l^0)\}$.

We call the location created by *Su* pattern, a success location. According with Definition 21, a success location is *not* deadlock if and only if there is a similar success location in another automaton.

The *Pfx* pattern prefixes a location to an automata.

Definition 24 (Prefix pattern). *Let* $A = (Loc', Loc^{u\prime}, l^{0\prime}, E', I')$ *be a* **TA**. *Let* $l^0 \notin Loc'$. *Let* $\ell \in L$, *and let* $R \subseteq \mathbb{C}$. *Then,*

$$Pfx(l^0, \ell, R, A) = (Loc' \cup \{l^0\}, Loc^{u\prime} \cup \{l^0\}, l^0, E, I')$$

where: $E = E' \cup \{l^0, \ell, \texttt{true}, R, l^{0\prime}\}$.

The *IC* pattern prefixes a location to set of automata.

Definition 25 (Internal Choice pattern). *For all* $i \in I$, *let* $A_i = (Loc_i, Loc_i^u, l_i^0, E_i, I_i)$ *be a* **TA**. *Let all* $g_i \in \mathcal{G}_\mathbb{C}$ *and let* $g \in \mathcal{G}_\mathbb{C}$. *Then:*

$$IC(l^0, g, \{(g_i, A_i) \mid i \in I\}) = (Loc, Loc^u, l^0, E, I)$$

where: $Loc = \bigcup_i Loc_i \cup \{l^0\}$,$Loc^u = \bigcup_i Loc_i^u$, $E = \bigcup_i E_i \cup \bigcup_i \{(l^0, \tau, g_i, \emptyset, l_i^0)\}$, $I = \bigcup_i I_i \cup \{(l^0, g)\}$.

The *EC* pattern prefixes a location to set of automata.

Definition 26 (External Choice pattern). *For all* $i \in I$, *let* $A_i = (Loc_i, Loc_i^u, l_i^0, E_i, I_i)$ *be a* **TA**. *Let* $g_i, g \in \mathcal{G}_\mathbb{C}$, $R_i \subseteq \mathbb{C}$ *and* $\ell_i \in L$. *Then:*

$$EC(l^0, g, \{(\ell_i, g_i, R_i, A_i) \mid i \in I\}) = (Loc, Loc^u, l^0, E, I)$$

where: $Loc = \bigcup_i Loc_i \cup \{l^0\}$, $Loc^u = \bigcup_i Loc_i^u$, $E = \bigcup_i E_i \cup \bigcup_i \{(l^0, \ell_i, g_i, R_i, l_i^0)\}$, $I = \bigcup_i I_i \cup \{(l^0, g)\}$.

B.4 Encoding timed session types in TA.

We now proceed to map a TST into a **TA**s. To define the mapping, we use the patterns defined in Appendix B.3.

The mapping exploits the similarities between the behaviour of automata edges and the rules to fire an action in TST: in both cases guards must be satisfied, and a set of clocks can be reset afterwards. Furthermore, the synchronization on automata channels is very similar to the [S-τ] rule for TSTs, with the only difference that in automata synchronization, !-edge and ?-edge are symmetric.

The mapping creates a location for every term in $\mathcal{S}$ that can be reached in $\rightarrow_D$, and also creates an edge for each move that can be performed by the single TST, so that, in the end, the moves performed by the network of the mapped automata are the same moves of $\rightarrow_D$. Moreover, since location names are in $\mathcal{S}$, configuration in the network are pretty much similar to configuration in $\rightarrow_D$.

To allow time elapse as desired, we use $\mathtt{rdy}(p)$ as an invariant. Although $\mathtt{rdy}(p)$ is a semantic object (i.e. a set of clock valuations), it can always be represented as a syntactic term (i.e. a guard)[18]. Generally, such an invariant can contain disjunction: this is the reason why we are admitting disjunctions in invariants.

In Definition 28 we present the transformation of a DENF into an automaton: the function $\mathcal{T}(X, D)$ transform every single defining equation in D into an automaton; all the resulting automata are then composed together into a single one, through Definition 27.

Definition 27 (Union of TAs). *For all $i \in I$, let $A_i = (Loc_i, Loc_i^u, l_i^0, E_i, I_i)$ be a* **TA**. *Let $l \in \bigcup_i Loc_i$. The union of all the A_i is an automata defined as:*

$$\bigsqcup_{i\in I}^{l} A_i = (\bigcup_i Loc_i, \bigcup_i Loc_i^u, l, \bigcup_i E_i, I)$$

where for all $\hat{l}$, $I(\hat{l}) = \bigwedge_i I_i(\hat{l})$.

Definition 28 (Mapping). *We define the mapping $\mathcal{T}$ from DENFs to* **TA***s as follows, where locations names are in $\mathcal{S}$.*

$$\mathcal{T}(X, D) = \bigsqcup_{d\in D}^{X} [\![d]\!]$$

and:

$$\begin{array}{lll}
[\![X \triangleq \bigoplus_{i\in I} \mathtt{!a_i}\{g_i, R_i\}.X_i]\!] & = & Pfx(X, \tau, \emptyset, \\
 & & \quad IC(\tau X, \Omega_d, \{(g_i, A_i)\}_i) \ \textit{where} \\
 & & A_i = Pfx([\mathtt{!a_i}\{g_i, R_i\}]\, X_i, \mathtt{!a_i}, R_i, Idle(X_i)) \\
[\![X \triangleq \sum_{i\in I} \mathtt{?a}\{g_i, R_i\}.X_i]\!] & = & EC(X, \Omega_d, \{(\mathtt{?a_i}, g_i, R_i, Idle(X_i))\}_i) \\
[\![X \triangleq \mathbf{1}]\!] & = & Su(X)
\end{array}$$

where $\Omega_{X \triangleq p} = \mathtt{rdy}(p)$.

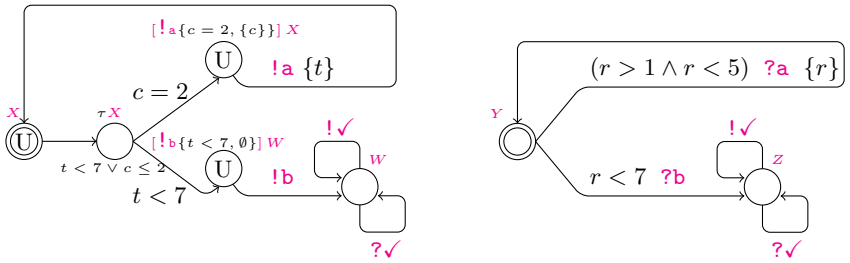

Fig. 8. TAs for Example 12

Example 12. Let $p_0 = \mathrm{rec}\, X_0.\, \texttt{!a}\{c = 2, \{c\}\}.X_0 \oplus \texttt{!b}\{t < 7, \emptyset\}$. By Definition 15, $\langle p_0 \rangle = (X, D)$ with $D = \{X \triangleq p, W \triangleq \mathbf{1}\})$ and

$$p = \texttt{!a}\{c = 2, \{c\}\}.X \oplus \texttt{!b}\{t < 7, \emptyset\}.W$$

The automaton resulting from transformation $\mathcal{T}(X, D)$ is depicted on the left of Figure 8. Its locations are: $\{X, W, \tau X, [\texttt{!a}\{c = 2, \{c\}\}]\, X, [\texttt{!b}\{t < 7, \emptyset\}]\, W\}$, with location τX and W being not urgent. Invariants are set to true, except for the one in τX. Indeed, since p is an internal choice, according with the mapping we have:

$$I(\tau X) = \Omega_{X \triangleq p} = \mathtt{rdy}(p) = \downarrow ([\![c = 2]\!] \cup [\![t < 7]\!]) \equiv (c \leq 2) \vee (t < 7)$$

Let $q_0 = \mathrm{rec}\, Y_0.\, \texttt{?a}\{r > 1 \wedge r < 5, \{r\}\}.Y_0 \oplus \texttt{?b}\{r < 7, \emptyset\}$. By Definition 15, $\langle q_0 \rangle = (Y, F)$ with $F = \{Y \triangleq q, Z \triangleq \mathbf{1}\})$ and

$$q = \texttt{?a}\{(r > 1 \wedge r < 5), \{r\}\}.Y + \texttt{?b}\{r < 7, \emptyset\}.Z$$

The automaton resulting from transformation $\mathcal{T}(Y, F)$ is depicted on the right of Figure 8. Its locations are: $\{Y, Z\}$. Invariants are all set to true. Indeed, since q is an external choice, according with the mapping we have:

$$I(Y) = \Omega_{Y \triangleq q} = \mathtt{rdy}(q) = \mathbb{V} \equiv \mathit{true}$$

Note that $p_0 \bowtie q_0$, and that the network $\langle p_0 \rangle \mid \langle q_0 \rangle$ is not deadlock.

B.5 Model checking compliance

In this section we prove that compliance between TSTs is decidable, by reducing it to the problem of checking if a network of **TA**s is not deadlock — which is known to be decidable.

Our reduction exploits the mapping from TSTs to**TA**s introduced in Definition 28. In Lemma 7 we will show that configurations of TSTs (in DENF) are strongly bisimilar to the network of **TA**s constructed by Definition 28

In Definition 29 we recall the definition of strong bisimulation.

Definition 29 (Strong bisimulation). *A binary relation $\mathcal{R}$ between states of an LTS is a bisimulation if whenever whenever $s_1 \mathcal{R} s_2$, and α is an action:*

1. *if $s_1 \xrightarrow{\alpha} s_1'$ then there is a transition $s_2 \xrightarrow{\alpha} s_2'$ such that $s_1' \mathcal{R} s_2'$*
2. *if $s_2 \xrightarrow{\alpha} s_2'$ then there is a transition $s_1 \xrightarrow{\alpha} s_1'$ such that $s_1' \mathcal{R} s_2'$*

Two states s_1 and s_2 are bisimilar, written $s_1 \sim s_2$, iff there is a bisimulation that relates them. Henceforth the relation $\sim$ will be referred to strong bisimulation equivalence or strong bisimilarity.

Given a set of defining equations D, we define $\mathtt{ck}(D)$ as the set of clocks used in D.

Lemma 7. *Let (X, D') and (Y, D'') be two DENF, with $\mathtt{rv}(D') \cap \mathtt{rv}(D'') = \emptyset = \mathtt{ck}(D') \cap \mathtt{ck}(D'')$. Let $A_1 = [\![D']\!]^X = (Loc_1, Loc_1^u, l_1^0, E_1, I_1)$ and $A_2 = [\![D'']\!]^Y = (Loc_2, Loc_2^u, l_2^0, E_2, I_2)$. Let $N = A_1 \mid A_2$. Then:*

$$((X, \nu_0)|(Y, \eta_0), \rightarrow_{D' \cup D''}) \sim ((X, Y, \nu_0 \cup \eta_0), \rightarrow_N)$$

Proof. Let (X, D') and (Y, D'') as in the statement. For all ν, η, we define the union of two clock valuations as:

$$(\nu \sqcup \eta)(t) = \begin{cases} \nu(t) & \text{if } t \in \mathtt{ck}(D') \\ \eta(t) & \text{if } t \in \mathtt{ck}(D'') \\ 0 & otherwise \end{cases} \tag{4}$$

Then we define the relation $\mathcal{R}$ as follows:

$$\mathcal{R} = \{((x, \nu)|(y, \eta), \quad (x, y, \nu \sqcup \eta)) \mid x, y, z \in \mathcal{S} \text{ and } \nu \in [\![I_1(x)]\!] \text{ and } \eta \in [\![I_2(y)]\!]\} \tag{5}$$

First, we show that every possible move from configuration $(x, \nu)|(y, \eta)$ in $\rightarrow_D$ is matched by a move from state $(x, y, \nu \cup \eta)$ in $\rightarrow_N$, and that the resulting states belong to $\mathcal{R}$. According to the rule of Definition 18 used to justify the move, we have the following cases:

[DES-$\oplus$]:

$$\frac{(x, \nu) \xrightarrow{\tau}_D (x', \nu)}{(x, \nu) \mid (y, \eta) \xrightarrow{\tau}_D (x', \nu) \mid (y, \eta)}$$

According to this rule, we have two sub cases in which the premise can fire a τ move:

[DE-*empty*]: $(x, \nu) \xrightarrow{\tau}_D (\tau X, \nu)$ with $x = X$ and $X \triangleq p \in D$ with $p = \oplus \ldots$ and $\nu \in \mathtt{rdy}(p)$. According with Definition 28, the mapping of an internal choice is $[\![X \triangleq p]\!] = Pfx(X, \tau, \emptyset, IC(\tau X, \mathtt{rdy}(p), \{(g_i, A_i)\}_i))$ for some A_i and g_i. Hence, according to Definition 24 of *Pfx* pattern, there exists an edge in E_1 such as $(X, \tau, \mathtt{true}, \emptyset, \tau X)$. According to Definition 25 of *IC* pattern, $I_1(\tau X) = \mathtt{rdy}(p)$. In our case rule [TA1] of Definition 20 states that:

$$\frac{(X, \tau, \mathtt{true}, \emptyset, \tau X) \in E_1 \qquad \nu \sqcup \eta \in [\![\mathtt{true}]\!] \qquad (\nu \sqcup \eta)[\emptyset] \in [\![I_1(\tau X) \wedge I_2(y)]\!]}{(X, y, \nu \sqcup \eta) \xrightarrow{\tau}_N (\tau X, y, \nu \sqcup \eta)}$$

We must show that $\nu \sqcup \eta \in [\![\texttt{true}]\!]$ and $\nu \sqcup \eta \in [\![I_1(x) \wedge I_2(y)]\!]$. The former holds trivially. For the second, we have $I_1(\tau X) = \texttt{rdy}(p)$; by premises in [DES-⊕] we have $\nu \in \texttt{rdy}(p)$ and, according to Equation (6) since $\texttt{ck}(D') \cap \texttt{ck}(D'') = \emptyset$, we have $\nu \sqcup \eta \in \texttt{rdy}(p)$. $\nu \sqcup \eta \in [\![I_2(y)]\!]$ holds by hypothesis in the definition of $\mathcal{R}$ in Equation (5). Hence $(X, y, \nu \sqcup \eta) \xrightarrow{\tau}_N (\tau X, y, \nu \sqcup \eta)$ and the resulting states belong to $\mathcal{R}$.

[DE-⊕]: $(x, \nu)|(y, \eta) \xrightarrow{\tau}_D (x', \nu)|(y, \eta)$ with $x = \tau X$ and $X \triangleq \texttt{!a}\{g, R\}.\, Y \oplus p'$ and $\nu \in [\![g]\!]$ and $x' = [\texttt{!a}\{g, R\}]\, Y$. According with Definition 28, the mapping of an internal choice is

$$[\![X \triangleq p]\!] = Pfx(X, \tau, \emptyset, IC(\tau X, \texttt{rdy}(p), \{(g, A)\} \cup S))$$

for some set S, and with $A = Pfx([\texttt{!a}\{g, R\}]\, Y, \texttt{!a}, R, Idle(Y))$. According to Definition 25 of IC pattern, there exists an edge in E_1 such as $(\tau X, \tau, g, \emptyset, x')$ with $I_1(x') = \texttt{true}$. In our case rule [TA1] of Definition 20 states that:

$$\frac{(\tau X, \tau, g, \emptyset, x') \in E_1 \qquad \nu \sqcup \eta \in [\![g]\!] \qquad (\nu \sqcup \eta)[\emptyset] \in [\![I_1(x') \wedge I_2(y)]\!]}{(\tau X, y, \nu \sqcup \eta) \xrightarrow{\tau}_N (x', y, \nu \sqcup \eta)}$$

We must show that $\nu \sqcup \eta \in [\![g]\!]$ and $\nu \sqcup \eta \in [\![I_1(x') \wedge I_2(y)]\!]$. The former holds by hypothesis since $\nu \in [\![g]\!]$. For the second, we have $I_1(\tau X) = \texttt{true}$– hence $\nu \sqcup \eta \in [\![I_1(x')]\!]$ is trivially true; and $\nu \sqcup \eta \in [\![I_2(y)]\!]$ holds by hypothesis in the definition of $\mathcal{R}$ in Equation (5). Hence $(\tau X, y, \nu \sqcup \eta) \xrightarrow{\tau}_N (x', y, \nu \sqcup \eta)$ and the resulting states belong to $\mathcal{R}$.

[DES-DEL]:

$$\frac{(x, \nu) \xrightarrow{\delta}_D (x, \nu') \quad (y, \eta) \xrightarrow{\delta}_D (y, \eta')}{(x, \nu) \mid (y, \eta) \xrightarrow{\delta}_D (x, \nu') \mid (y, \eta')}$$

According to this rule, we have several possible combinations of the premises to fire a δ move:

[DE-DEL2] applied twice:

$$\frac{\dfrac{X \triangleq p \in D \quad \nu + \delta \in \texttt{rdy}(p)}{(\tau X,\ \nu) \xrightarrow{\delta}_D (\tau X,\ \nu + \delta)} \qquad \dfrac{Y \triangleq q \in D \quad \nu + \delta \in \texttt{rdy}(q)}{(\tau Y,\ \nu) \xrightarrow{\delta}_D (\tau Y,\ \nu + \delta)}}{(\tau X, \nu) \mid (\tau Y, \eta) \xrightarrow{\delta}_D (\tau X, \nu + \delta) \mid (\tau Y, \eta + \delta)}$$

According with Definition 28, there is only a case in which a location name has prefix τ, and in that case we have $I_1(\tau X) = \texttt{rdy}(p)$ and $\tau X \notin Loc_1^u$. Similarly, $I_2(\tau Y) = \texttt{rdy}(q)$ and $\tau Y \notin Loc_2^u$. In our case rule [TA3] of Definition 20 states that:

$$\frac{((\nu \sqcup \eta) + \delta) \in [\![I_1(\tau X) \wedge I_2(\tau Y)]\!] \qquad \tau X \notin Loc_1^u \qquad \tau Y \notin Loc_2^u}{(\tau X, \tau Y, \nu \sqcup \eta) \xrightarrow{\delta}_N (\tau X, \tau Y, (\nu \sqcup \eta) + \delta)}$$

By [DE-DEL2] rule, we have $\nu+\delta \in \mathtt{rdy}(p)$, hence, since $\mathtt{ck}(D') \cap \mathtt{ck}(D'') = \emptyset$ and by Equation (6), it follows $(\nu \sqcup \eta) + \delta \in \mathtt{rdy}(p)$. Similarly, since $\eta + \delta \in \mathtt{rdy}(q)$, it follows $(\nu \sqcup \eta) + \delta \in \mathtt{rdy}(q)$. We already noted that τY and τX are not urgent, so we conclude $(\tau X, y, \nu \sqcup \eta) \xrightarrow{\delta}_N (\tau X, y, \nu + \delta \sqcup \eta + \delta)$. The resulting states belong to $\mathcal{R}$.

[DE-DEL2] and [DE-DEL1]:

$$\frac{(X \triangleq \ldots) \in D \vee (X \triangleq \mathbf{1}) \in D}{(X,\ \nu) \xrightarrow{\delta}_D (X,\ \nu+\delta)} \qquad \frac{Y \triangleq q \in D \quad \nu+\delta \in \mathtt{rdy}(q)}{(\tau Y,\ \nu) \xrightarrow{\delta}_D (\tau Y,\ \nu+\delta)}$$
$$(X,\nu) \mid (\tau Y,\eta) \xrightarrow{\delta}_D (X,\nu+\delta) \mid (\tau Y,\eta+\delta)$$

According with Definition 28, there is only a case in which a location name has prefix τ, and in that case we have $I_2(\tau Y) = \mathtt{rdy}(q)$ and $\tau Y \notin Loc_2^u$. For X we have two possibilities: either (i) it is an internal choice or (i) it is a success term.

(i) In the first case, according with Definition 28, the mapping for an external choice is: $[\![X \triangleq p]\!] = EC(X, \mathtt{rdy}(p), S)$ for some set S. Since $\mathtt{rdy}(p)$ is $\mathtt{true}$ for external choices, then there exists a location X with invariant $I_1(X) = \mathtt{rdy}(p) = \mathtt{true}$ and $\tau X \notin Loc_1^u$. In our case rule [TA3] of Definition 20 states that:

$$\frac{((\nu \sqcup \eta) + \delta) \in [\![\mathtt{true} \wedge I_2(\tau Y)]\!] \qquad X \notin Loc_1^u \qquad \tau Y \notin Loc_2^u}{(X, \tau Y, \nu \sqcup \eta) \xrightarrow{\delta}_N (X, \tau Y, (\nu \sqcup \eta) + \delta)}$$

By [DE-DEL2] rule, we have $\eta + \delta \in \mathtt{rdy}(q)$, it follows $(\nu \sqcup \eta) + \delta \in \mathtt{rdy}(q)$. We already noted that τY and X are not urgent, so we conclude $(X, y, \nu \sqcup \eta) \xrightarrow{\delta}_N (X, y, \nu + \delta \sqcup \eta + \delta)$. The resulting states belong to $\mathcal{R}$.

(ii) In the second case, according with Definition 28, the mapping for the success termination is: $[\![X \triangleq \mathbf{1}]\!] = Su(X)$. Hence, according to Definition 23 of Su pattern, there exists an edge $(X, !\checkmark, \mathtt{true}, \emptyset, X) \in E_1$ with $I_1(X) = \mathtt{true}$ and $\tau X \notin Loc_1^u$. In our case rule [TA3] of Definition 20 states that:

$$\frac{((\nu \sqcup \eta) + \delta) \in [\![\mathtt{true} \wedge I_2(\tau Y)]\!] \qquad X \notin Loc_1^u \qquad \tau Y \notin Loc_2^u}{(X, \tau Y, \nu \sqcup \eta) \xrightarrow{\delta}_N (X, \tau Y, (\nu \sqcup \eta) + \delta)}$$

By IDRULE rule, we have $\eta + \delta \in \mathtt{rdy}(q)$, it follows $(\nu \sqcup \eta) + \delta \in \mathtt{rdy}(q)$. We already noted that τY and X are not urgent, so we conclude $(X, y, \nu \sqcup \eta) \xrightarrow{\delta}_N (X, y, \nu + \delta \sqcup \eta + \delta)$. The resulting states belong to $\mathcal{R}$.

[DE-DEL1] applied twice: Similar to previous cases.

[DES-τ]:

$$\frac{(x,\nu) \xrightarrow{!a}_D (x',\nu') \quad (y,\eta) \xrightarrow{?a}_D (y',\eta')}{(x,\nu) \mid (y,\eta) \xrightarrow{a}_D (x',\nu') \mid (y',\eta')}$$

According to this rule, we can synchronize on $\mathtt{a}$ only if

$$\frac{([\,!\mathtt{a}\{g,R\}]\,X',\ \nu)\ \xrightarrow{!\mathtt{a}}_D\ (X',\ \nu[R]) \qquad \dfrac{(Y \triangleq\ ?\mathtt{a}\{f,T\}.\,Y'+q) \in D \quad \eta \in [\![f]\!]}{(Y,\ \nu)\ \xrightarrow{?\mathtt{a}}_D\ (Y',\ \eta[T])}}{([\,!\mathtt{a}\{g,R\}]\,X',\nu)\mid(Y,\eta)\xrightarrow{\mathtt{a}}_D(X',\nu[R])\mid(Y',\eta[T])}$$

Let $x = [\,!\mathtt{a}\{g,R\}]\,X'$. According with Definition 28, the last part of the mapping for an internal choice is such as $Pfx(x,!\mathtt{a},R,Idle(X'))$. Hence, according to Definition 24 of Pfx pattern, there exists an edge $(x,!\mathtt{a},\mathtt{true},R,X') \in E_1$ and $I_1(X') = \mathtt{true}$. According with Definition 28, the mapping for an external choice is: $[\![Y \triangleq\ ?\mathtt{a}\{f,T\}.\,Y'+q]\!] = EC(Y,\mathtt{true},\{(?\mathtt{a},f,T,Idle(Y'))\}\cup S)$ for some S. Hence, according to Definition 26 of EC pattern, there exists an edge $(Y,?\mathtt{a},f,T,Y')$ and $I_1(Y') = \mathtt{true}$. In our case rule [TA2] of Definition 20 states that:

$$\frac{\begin{array}{l}(x,!\mathtt{a},\mathtt{true},R,X') \in E_1 \quad (\nu \sqcup \eta) \in [\![\mathtt{true}]\!] \\ (Y,?\mathtt{a},f,T,Y') \in E_2 \quad (\nu \sqcup \eta) \in [\![f]\!]\end{array} \qquad (\nu \sqcup \eta)[R][T] \in [\![\mathtt{true} \wedge \mathtt{true}]\!]}{(x,Y,\nu \sqcup \eta) \xrightarrow{\mathtt{a}}_N (X',Y',(\nu \sqcup \eta)[R][T])}$$

Since by hypothesis $\eta \in [\![f]\!]$, since $\mathtt{ck}(D') \cap \mathtt{ck}(D'') = \emptyset$ and by Equation (6) we obtain $(\eta \sqcup \nu) \in [\![f]\!]$. Hence by rule [TA2] of Definition 20 , we have $(x,Y,\nu \sqcup \eta) \xrightarrow{\mathtt{a}}_N (X',Y',\nu \sqcup \eta[R][T]$. The resulting states, belong to $\mathcal{R}$.

[DES-✓]:

$$\frac{X \triangleq \mathbf{1} \in D \quad Y \triangleq \mathbf{1} \in D}{(X,\nu)\mid(Y,\eta)\xrightarrow{\checkmark}_D(X,\nu)\mid(Y,\eta)}$$

According with Definition 28, we have $[\![X \triangleq \mathbf{1}]\!] = Su(X)$. Hence, according to Definition 23 of Su pattern, there exists an edge $(X,!\checkmark,\mathtt{true},\emptyset,X) \in E_1$ with $I_1(X) = \mathtt{true}$. Similarly, there exists an edge $(Y,?\checkmark,\mathtt{true},\emptyset,Y) \in E_2$ with $I_2(Y) = \mathtt{true}$. With a reasoning similar to the previous case, we can conclude that by rule [TA2] of Definition 20 we have $(X,Y,\nu \sqcup \eta) \xrightarrow{\checkmark}_D (X,Y,\nu \sqcup \eta)$. The resulting states, belong to $\mathcal{R}$.

Secondly, we show that every possible move from configuration $(x,y,\nu \sqcup \eta)$ in $\rightarrow_N$ is matched by a move from state $(x,\nu)|(y,\eta)$ in $\rightarrow_D$, and that the resulting states belong to $\mathcal{R}$. According to the rules of Definition 20 used to justify the move, we have the following cases:

[TA1]:

$$\frac{(x,\tau,g,R,x') \in E_1 \qquad (\nu \sqcup \eta) \in [\![g]\!] \qquad (\nu \sqcup \eta)[R] \in [\![I_1(x') \wedge I_2(y)]\!]}{(x,y,\nu \sqcup \eta) \xrightarrow{\tau}_N (x',y,\nu \sqcup \eta[R])}$$

According with Definition 28, there exist two possibilities for an edge in network N to display a τ label, and both derive from the mapping of an internal

choice. Let $p = \texttt{!a}\{g, R\}Y \oplus p'$, then $[\![X \triangleq p]\!] = \mathit{Pfx}(X, \tau, \emptyset, \mathit{IC}(\tau X, \texttt{rdy}(p), \{(g, A)\} \cup S))$ for some set S and automaton A. Hence, according to Definition 24 and Definition 25 of patterns, we have two edges with τ label in E_1: (i) $(X, \tau, \texttt{true}, \emptyset, \tau X)$ with $I_1(\tau X) = \texttt{rdy}(p)$; and (ii) $(\tau X, \tau, g, \emptyset, Y)$ with $I_1(x') = \texttt{true}$. So we have two sub-cases:

(i) Let assume we fire the first edge $(X, \tau, \texttt{true}, \emptyset, \tau X)$. Hence $x = X$, $x' = \tau X$ and $R = \emptyset$. By hypothesis $\nu \sqcup \eta \in [\![I_1(\tau X)]\!] = \texttt{rdy}(p)$. By Equation (6) since $\texttt{ck}(D') \cap \texttt{ck}(D'') = \emptyset$, we derive $\nu \in \texttt{rdy}(p)$. Hence, we can use [DE-*empty*] and [DES-$\oplus$] rule from Definition 18 to obtain:

$$\frac{(X \triangleq p) \in D \quad p = \oplus \ldots \quad \nu \in \texttt{rdy}(p)}{(X,\ \nu) \xrightarrow{\tau}_D (\tau X,\ \nu)} \qquad \frac{}{(X, \nu) \mid (y, \eta) \xrightarrow{\tau}_D (\tau X, \nu) \mid (y, \eta)}$$

Since by [TA1], $(\nu \sqcup \eta)[\emptyset] \in [\![I_1(\tau X) \wedge I_2(y)]\!]$, the resulting states, belong to $\mathcal{R}$.

(ii) Let assume we fire the second edge $(\tau X, \tau, g, \emptyset, x')$. Hence $x = \tau X$, $x' = Y$ and $R = \emptyset$. By [TA1], $\nu \sqcup \eta \in [\![g]\!]$, by Equation (6) since $\texttt{ck}(D') \cap \texttt{ck}(D'') = \emptyset$, we derive $\nu \in [\![g]\!]$. Hence, we can use [DE-$\oplus$] and [DES-$\oplus$] rules from Definition 18 to obtain:

$$\frac{(X \triangleq \texttt{!a}\{g, R\}Y \oplus p') \in D \quad \nu \in [\![g]\!]}{(\tau X,\ \nu) \xrightarrow{\tau}_D ([\texttt{!a}\{g, R\}]\, Y,\ \nu)} \qquad \frac{}{(\tau X, \nu) \mid (y, \eta) \xrightarrow{\tau}_D ([\texttt{!a}\{g, R\}]\, Y, \nu) \mid (y, \eta)}$$

Since by hypothesis, $(\nu \sqcup \eta)[\emptyset] \in [\![I_1(x') \wedge I_2(y)]\!]$, the resulting states, belong to $\mathcal{R}$.

[TA2]:

$$\frac{(x, \texttt{!a}, g, R, x') \in E_1 \quad (\nu \sqcup \eta) \in [\![g]\!] \quad (\nu \sqcup \eta)[R][T] \in [\![I_1(x)]\!] \quad (y, \texttt{?a}, f, T, y') \in E_2 \quad (\nu \sqcup \eta) \in [\![f]\!] \quad (\nu \sqcup \eta)[R][T] \in [\![I_2(y)]\!]}{(x, y, \nu \sqcup \eta) \xrightarrow{\texttt{a}}_N (x', y', (\nu \sqcup \eta)[R][T])}$$

According with Definition 28, there exist two possibilities for a synchronization to happen: either (i) for an internal-external choice synchronization, or (ii) because both automata reached a location generated by *Su* pattern. Hence we have two sub-cases:

(i) If the label-firing is due to an internal-external choice synchronization, then we have $x = [\texttt{!a}\{g, R\}]\, X$ and $y = Y$ for some $Y \triangleq \texttt{?a}\{f, S\}.Y' + p$. According with Definition 28, the last part of the mapping for an internal choice is such as $\mathit{Pfx}(x, \texttt{!a}, R, \mathit{Idle}(X'))$. Hence, according to Definition 24 of *Pfx* pattern, there exists an edge $(x, \texttt{!a}, \texttt{true}, R, X') \in E_1$ and $I_1(X') = \texttt{true}$. According with Definition 28, the mapping for an external choice is: $[\![Y \triangleq \texttt{?a}\{f, T\}.Y' + q]\!] = \mathit{EC}(Y, \texttt{true}, \{(\texttt{?a}, f, T, \mathit{Idle}(Y'))\} \cup$

S) for some S. Hence, according to Definition 26 of EC pattern, there exists an edge $(Y, \texttt{?a}, f, T, Y')$ and $I_1(Y') = \texttt{true}$. So in this case [TA2] becomes:

$$\frac{\begin{array}{c}(x, \texttt{!a}, \texttt{true}, R, X') \in E_1 \quad (\nu \sqcup \eta) \in [\![\texttt{true}]\!] \quad (\nu \sqcup \eta)[R][T] \in [\![I_1(X')]\!] \\ (Y, \texttt{?a}, f, T, Y') \in E_2 \quad (\nu \sqcup \eta) \in [\![f]\!] \quad (\nu \sqcup \eta)[R][T] \in [\![I_2(Y')]\!]\end{array}}{(x, Y, \nu \sqcup \eta) \xrightarrow{\texttt{a}}_N (X', Y', (\nu \sqcup \eta)[R][T])}$$

Since $\texttt{ck}(D') \cap \texttt{ck}(D'') = \emptyset$ and since $(\nu \sqcup \eta) \in [\![f]\!]$, by Equation (6), we derive $\eta \in [\![f]\!]$. Hence we can apply [DE-?][DE-!] and [DES-τ] to obtain:

$$\frac{([\texttt{!a}\{g, R\}]\, X',\ \nu) \xrightarrow{\texttt{!a}}_D (X',\ \nu[R]) \quad \dfrac{(Y \triangleq \texttt{?a}\{f, T\}.\, Y' + q) \in D \quad \eta \in [\![f]\!]}{(Y,\ \nu) \xrightarrow{\texttt{?a}}_D (Y',\ \eta[T])}}{([\texttt{!a}\{g, R\}]\, X', \nu) \mid (Y, \eta) \xrightarrow{\texttt{a}}_D (X', \nu[R]) \mid (Y', \eta[T])}$$

Since by hypothesis, $(\nu \sqcup \eta)[R][T] \in [\![I_1(X) \wedge I_2(Y)]\!]$, the resulting states belong to $\mathcal{R}$.

(ii) If the label-firing is due to both automata reaching Su pattern state, then we have $x = X$ and $X \triangleq \mathbf{1}$ and $y = Y$ and $Y \triangleq \mathbf{1}$. There exists an edge on the first automata such as $(X, \texttt{!}\checkmark, \mathit{true}, \emptyset, X)$; and an edge on the second automata such as $(Y, \texttt{?}\checkmark, \mathit{true}, \emptyset, Y)$.
So in this case [TA2] becomes:

$$\frac{\begin{array}{c}(X, \texttt{!}\checkmark, \texttt{true}, \emptyset, X) \in E_1 \quad (\nu \sqcup \eta) \in [\![\texttt{true}]\!] \quad (\nu \sqcup \eta) \in [\![I_1(X)]\!] \\ (Y, \texttt{!}\checkmark, \texttt{true}, \emptyset, Y) \in E_2 \quad (\nu \sqcup \eta) \in [\![\texttt{true}]\!] \quad (\nu \sqcup \eta) \in [\![I_2(Y)]\!]\end{array}}{(X, Y, \nu \sqcup \eta) \xrightarrow{\checkmark}_N (X, Y, (\nu \sqcup \eta)}$$

Hence, we can apply [DES-$\checkmark$] to obtain:

$$\frac{X \triangleq \mathbf{1} \in D \quad Y \triangleq \mathbf{1} \in D}{(X, \nu) \mid (Y, \eta) \xrightarrow{\checkmark}_D (X, \nu) \mid (Y, \eta)}$$

Since by hypothesis, $(\nu \sqcup \eta) \in [\![I_1(X) \wedge I_2(Y)]\!]$, the resulting states, belong to $\mathcal{R}$.

[TA3]:

$$\frac{((\nu \sqcup \eta) + \delta) \in [\![I_1(x)]\!] \quad x \notin \mathit{Loc}_1^u \quad ((\nu \sqcup \eta) + \delta) \in [\![I_2(y)]\!] \quad y \notin \mathit{Loc}_2^u}{(x, y, \nu \sqcup \eta) \xrightarrow{\delta}_N (x, y, (\nu \sqcup \eta) + \delta)}$$

According to [TA3], time can pass in a network only if all the locations of the current state are not urgent. According with Definition 28, this happens for: the second location in the mapping of an internal choice, the first location in the mapping of an external choice and the Su pattern location. Hence, we have several sub-cases:

(i) Let us assume that x derive from an internal choice and y from an external choice. According with Definition 28, x must be of the form $x = I_1(\tau X)$ for some $X \triangleq p \in D$ with $p = \oplus\ldots$, obtaining $I_1(\tau X) = \mathtt{rdy}(p)$ and $\tau X \notin Loc_1^y$. According with Definition 28, y must be of the form $y = Y$ for some $Y \triangleq q \in D$ with $q = +\ldots$, obtaining $I_2(Y) = \mathtt{rdy}(q) = \mathtt{true}$ and $\tau Y \notin Loc_2^y$. So in this case [TA2] becomes:

$$\frac{((\nu \sqcup \eta) + \delta) \in [\![I_1(\tau X)]\!] \quad (\tau X) \notin Loc_1^y \quad ((\nu \sqcup \eta) + \delta) \in [\![I_2(Y)]\!] \quad (Y) \notin Loc_2^y}{(l_1, l_2, \nu \sqcup \eta) \xrightarrow{\delta}_N (l_1, l_2, (\nu \sqcup \eta) + \delta)}$$

By hypothesis of [TA2], $((\nu \sqcup \eta) + \delta) \in [\![I_1(\tau X)]\!] = \mathtt{rdy}(p)$. Since $\mathtt{ck}(D') \cap \mathtt{ck}(D'') = \emptyset$, by Equation (6), we derive $\nu + \delta \in \mathtt{rdy}(p)$. Hence, we have:

$$\frac{\dfrac{X \triangleq p \in D \quad \nu + \delta \in \mathtt{rdy}(p)}{(\tau X,\ \nu) \xrightarrow{\delta}_D (\tau X,\ \nu + \delta)} \quad \dfrac{(Y = +\ldots) \in D \vee (Y = \mathbf{1}) \in D}{(Y,\ \nu) \xrightarrow{\delta}_D (Y,\ \nu + \delta)}}{(\tau X, \nu) \mid (Y, \eta) \xrightarrow{\delta}_D (\tau X, \nu + \delta) \mid (Y, \eta + \delta)}$$

Since by hypothesis, $((\nu \sqcup \eta) + \delta) \in [\![I_1(\tau X) \wedge I_2(Y)]\!]$, the resulting states, belong to $\mathcal{R}$.

(ii) All the other combinations can be proved similarly to the previous one.

Theorem 10. *Let (X, D') and (Y, D'') be two DENF, with $\mathtt{rv}(D') \cap \mathtt{rv}(D'') = \emptyset = \mathtt{ck}(D') \cap \mathtt{ck}(D'')$. Then:*

$$X \bowtie_D Y \iff [\![X]\!] \mid [\![Y]\!] \text{ is not deadlock}$$

Proof. Let (X, D') and (Y, D'') as in the statement. Let $D = D' \cup D''$. For all ν, η, we define the union of two clock valuations as:

$$(\nu \sqcup \eta)(t) = \begin{cases} \nu(t) & \text{if } t \in \mathtt{ck}(D') \\ \eta(t) & \text{if } t \in \mathtt{ck}(D'') \\ 0 & otherwise \end{cases} \tag{6}$$

$\Rightarrow$) Suppose, by absurd, that $X \bowtie Y$ but $[\![X]\!] \mid [\![Y]\!]$ is deadlock. Hence, according with Definition 21, there exists a trace λ such that $(X, Y, \nu_0 \sqcup \eta_0) \xrightarrow{\lambda}_N (x, y, \nu \sqcup \eta) = s$ and there is no δ nor $\mathtt{a}_\tau \in \mathtt{A} \cup \{\tau\}$ such that $s \xrightarrow{\delta}_N \xrightarrow{\mathtt{a}_\tau}_N s'$. By Lemma 7, $(X, \nu_0) | (Y, \eta_0) \xrightarrow{\lambda}_D (x, \nu) | (y, \eta) = c$. Since $X \bowtie_D Y$, by Definition 19, c can fire a move but by Lemma 7 we have that also s can fire that move. Which lead us to a contradiction.

$\Leftarrow$) Similar to the previous case.

Theorem 11. *Let p and q be two TSTs with no shared clocks and no free variables. Then:*

$$p \bowtie q \iff [\![\langle p \rangle]\!] \mid [\![\langle q \rangle]\!] \text{ is not deadlock}$$

Proof. Trivial by Lemma 6 and Theorem 11.

B.6 Model checking compliance in UPPAAL

The state reachability problem for timed automata is decidable, and methods and tools for checking both safety and liveness properties have been developed and intensively studied over the last 20 years.

Uppaal [7] is a model-checker for **TA**, with a modeling language and verification algorithms. The property of *deadlock-freeness* of a network, can be verified in Uppaal through its query language, which is a simplified version of Time Computation Tree Logic. A network N is deadlock-free if no one of the reachable states of the system is a deadlock state. The deadlock property is checked using the keyword `deadlock` in the path formula `A[] not deadlock` and it is satisfied if every reachable state in $\rightarrow_N$ is not a deadlock state.

Definition 30. *Let $N = A_1 \mid A_2$ be a network of* **UTA***s. Then, N is not deadlock iff the query* `A[] not deadlock` *is satisfied.*

The automata resulting from the mapping of Definition 28 allow for disjunctions in both guards and invariants; and their presence is known not to be an hindrance to decidability. However, disjunctions have not been implemented in *Uppaal* because of efficiency issues, so in order to actually implement our mapping with **UTA**s, we need to do some minor adjustments.

Instead of a disjunction on an edge, we can have several copies of the edge, one for each disjunct. For instance, the mapping of $\texttt{!a}\{t < 4 \vee t > 10\}$ may give an automata with edges $(l_0, \texttt{!a}, t < 4, \emptyset, l_1)$ and $(l_0, \texttt{!a}, t > 10, \emptyset, l_1)$ for some l_0 and l_1. Similarly, in case a location has an invariant with disjunctions, the invariant can be split into its disjuncts, and the location can be multiplied as needed. Finally, the *urgent* locations we use in Appendix B.2 are exactly those defined in **UTA**.

C Proofs for Section 3

C.1 Coinductive characterisation of compliance

Lemma 8. *Let $\mathcal{R}$ be a coinductive compliance relation, and let $(p, \nu)\ \mathcal{R}\ (q, \eta)$. Then:*

1. $(p, \nu) \mid (q, \eta)$ *not deadlock*
2. $(p, \nu) \mid (q, \eta) \xrightarrow{\tau} \gamma \implies \exists!\ p', q', \nu', \eta' : \gamma \xrightarrow{\tau} (p', \nu') \mid (q', \eta') \wedge (p', \nu')\ \mathcal{R}\ (q', \eta')$
3. $(p, \nu) \mid (q, \eta) \xrightarrow{\delta} (p', \nu') \mid (q', \eta') \implies (p', \nu')\ \mathcal{R}\ (q', \eta')$

Proof. First note that $\mathcal{R}$ does not talk about committed choices (i.e. terms of the form $[\texttt{!a}\{g, R\}]\, p$). Hence, assume $(p, \nu)\ \mathcal{R}\ (q, \eta)$. We proceed by cases on the form of p (modulo unfolding of recursion). If $p = \mathbf{1}$, then by Definition 7 also $q = \mathbf{1}$ and so all items are trivial. We show the case for internal choices: External ones are similar.
Assume $p = \bigoplus_{i \in I} \texttt{!a}_i\{g_i, T_i\}\,.\,p_i$, by Definition 7 we have $q = \sum_{j \in J} \texttt{?a}_j\{g_j, T_j\}\,.\,q_j$.

Take some δ and i such that $\nu + \delta \in [\![g_i]\!]$. Their existence is guaranteed by Definition 7. Let $\nu' = \nu + \delta$ and $\eta' = \eta + \delta$. Then:

$$\dfrac{\dfrac{\nu' \in [\![g_i]\!]}{(\bigoplus_{i\in I} \mathtt{!a}_i\{g_i, T_i\}\,.\,p_i, \nu') \xrightarrow{\tau} ([\mathtt{!a}_i\{g_i, T_i\}]\,p_i, \nu')}\,[\oplus]}{(\bigoplus_{i\in I} \mathtt{!a}_i\{g_i, T_i\}\,.\,p_i, \nu') \mid (q, \eta') \xrightarrow{\tau} ([\mathtt{!a}_i\{g_i, T_i\}]\,p_i,\ \nu') \mid (q, \eta')}\,[\text{S-}\oplus] \tag{7}$$

Hence, $(p, \nu) \mid (q, \eta)$ is not deadlock, which proves item 1.

For item 2, assume $(p, \nu) \mid (q, \eta) \xrightarrow{\tau} \gamma$. The derivation of such step must be as in eq. (7), with $\delta = 0$ and $\gamma = ([\mathtt{!a}_i\{g_i, T_i\}]\,p_i,\ \nu) \mid (q, \eta)$. By Definition 7, there exists $j \in J$ such that $\mathtt{a}_i = \mathtt{a}_j$, $\eta \in [\![g_j]\!]$ and $p_i\ \mathcal{R}\ q_j$.Hence:

$$\dfrac{\dfrac{}{([\mathtt{!a}_i\{g_i, T_i\}]\,p_i,\ \nu) \xrightarrow{\mathtt{!a}_i} (p_i, \nu[T_i])}\,[!] \qquad \dfrac{\eta \in [\![g_j]\!]}{(q, \eta) \xrightarrow{\mathtt{?a}_j} (q_j, \eta[T_j])}\,[?]}{\gamma \xrightarrow{\tau} (p_i, \nu[T_i]) \mid (q_j, \eta[T_j])}\,[\text{S-}\tau]$$

Note that the destination state is unique because atoms in choices are pairwise distinct.

For item 3, assume $(p, \nu) \mid (q, \eta) \xrightarrow{\delta} (p', \nu') \mid (q', \eta')$. The derivation of this step must be as follows:

$$\dfrac{\dfrac{\nu + \delta \in \mathtt{rdy}(p) = \downarrow \bigcup [\![g_i]\!]}{(p, \nu) \xrightarrow{\delta} (p, \nu + \delta)}\,[\textsc{Del}] \qquad \dfrac{\eta + \delta \in \mathtt{rdy}(q) = \mathbb{V}}{(q, \eta) \xrightarrow{\delta} (q, \eta + \delta)}\,[\textsc{Del}]}{(p, \nu) \mid (q, \eta) \xrightarrow{\delta} (p, \nu + \delta) \mid (q, \eta + \delta)}\,[\text{S-}\textsc{Del}] \tag{8}$$

Let $\nu' = \nu + \delta$ and $\eta' = \eta + \delta$. We have to show $(p, \nu')\ \mathcal{R}\ (q, \eta')$. By eq. (8) we have that $\nu' \in \mathtt{rdy}(p)$. It remains to show:

$$\nu' + \delta' \in [\![g_i]\!] \implies \exists j : \mathtt{a}_i = \mathtt{a}_j \wedge \eta' + \delta' \in [\![g_j]\!] \wedge (p_i, \nu' + \delta'[T_i])\,\mathcal{R}\,(q_j, \eta' + \delta'[T_j])$$

The thesis follows by the assumption $(p, \nu)\ \mathcal{R}\ (q, \eta)$.

Lemma 9. *For all $(p, \nu), (q, \eta)$ such that $(p, \nu) \bowtie (q, \eta)$:*

$$p = \bigoplus_{i\in I} \mathtt{!a}_i\{g_i, R_i\}\,.\,p_i \implies q = \sum_{j\in J} \mathtt{?a}_j\{g_j, R_j\}\,.\,p_j$$

and

$$p = \sum_{i\in I} \mathtt{?a}_i\{g_i, R_i\}\,.\,p_i \implies q = \bigoplus_{j\in J} \mathtt{!a}_j\{g_j, R_j\}\,.\,p_j$$

Proof. Trivial.

Lemma 1. $p \bowtie q \iff \exists\, \mathcal{R}$ *coinductive compliance* $:\ (p, \nu_0)\ \mathcal{R}\ (q, \eta_0)$

Proof. We prove the more general statement:

$$(p,\nu) \bowtie (q,\eta) \iff \exists\, \mathcal{R} \text{ coinductive compliance } :\ (p,\nu)\ \mathcal{R}\ (q,\eta)$$

For the "only if" part we proceed by showing that $\bowtie$ is a coinductive compliance relation. Assume $(p,\nu)\bowtie(q,\eta)$. We show the case where $p = \bigoplus_{i\in I} \mathtt{!a}_i\{g_i, T_i\}\,.\,p_i$ (the case of external choice is similar and the case $p = \mathbf{1}$ is trivial). By Lemma 9, $q = \sum_{j\in J} \mathtt{?a}_j\{g_j, T_j\}\,.\,q_j$. Since $(p,\nu) \mid (q,\eta)$ is not deadlock, it must be $\nu \in \mathtt{rdy}(p)$. By Definition 7, it remains to show, for all δ, i:

$$\nu+\delta \in [\![g_i]\!] \implies \exists j :\ \mathtt{a}_i = \mathtt{a}_j \wedge \eta+\delta \in [\![g_j]\!] \wedge (p_i, (\nu+\delta)[T_i]) \bowtie (q_j, (\eta+\delta)[T_j])$$

Suppose, by contradiction, this is not the case, and take a δ and an i such that

$$\nu+\delta \in [\![g_i]\!] \wedge (\forall j :\ \mathtt{a}_i \neq \mathtt{a}_j \vee \eta+\delta \notin [\![g_j]\!] \vee (p_i, (\nu+\delta)[T_i]) \not\bowtie (q_j, (\eta+\delta)[T_j])$$

There are two cases; If $\delta = 0$:

$$(p,\nu) \mid (q,\eta) \xrightarrow{\tau} ([\mathtt{!a}_i\{g_i, T_i\}]\, p_i, \nu) \mid (q,\eta) = \gamma$$

If, for all j, $\mathtt{a}_i \neq \mathtt{a}_j \vee \nu+\delta \notin [\![g_j]\!]$, then γ is deadlock. Otherwise:

$$\gamma \xrightarrow{\tau} (p_i, (\nu+\delta)[T_i]) \mid (q_j, (\eta+\delta)[T_j]) = \gamma'$$

with $(p_i, (\nu+\delta)[T_i]) \not\bowtie (q_j, (\eta+\delta)[T_j])$, and so, by Definition 6, there exists some deadlock configuration γ'' such that $\gamma' \to^* \gamma''$. Hence $(p,\nu) \not\bowtie (q,\eta)$. If $\delta > 0$:

$$(p,\nu) \mid (q,\eta) \xrightarrow{\delta} (p, \nu+\delta) \mid (q, \eta+\delta)$$

The thesis follows by an argument similar to the case with $\delta = 0$.

The "if" part is a straightforward consequence of Lemma 8

C.2 Kind system

The following lemmata state that rule [Rec] is a g.f.p., which can be computed in a finite number of steps.

Lemma 10. *For all sets of clocks T and for all $\mathcal{K}, \mathcal{K}'$ such that $\mathcal{K} \subseteq \mathcal{K}'$:*

$$\downarrow \mathcal{K} \subseteq \downarrow \mathcal{K}' \text{ and } \mathcal{K}[T]^{-1} \subseteq \mathcal{K}'[T]^{-1}$$

Proof. Assume $\mathcal{K} \subseteq \mathcal{K}'$. Then:

$$\downarrow \mathcal{K} = \{\nu \mid \exists \delta : \nu+\delta \in \mathcal{K}\} \subseteq \{\nu \mid \exists \delta : \nu+\delta \in \mathcal{K}'\} = \downarrow \mathcal{K}'$$

and

$$\mathcal{K}[T]^{-1} = \{\nu \mid \nu[T] \in \mathcal{K}\} \subseteq \{\nu \mid \nu[T] \in \mathcal{K}'\} = \mathcal{K}'[T]^{-1}$$

Corollary 1. *For all g, T and for all $\mathcal{K}, \mathcal{K}'$ such that $\mathcal{K} \subseteq \mathcal{K}'$:*

1.
$$\downarrow \left([\![g]\!] \cap \mathcal{K}[T]^{-1} \right) \subseteq \downarrow \left([\![g]\!] \cap \mathcal{K}'[T]^{-1} \right)$$

2.
$$\left(\downarrow [\![g]\!] \right) \setminus \left(\downarrow \left([\![g]\!] \setminus \mathcal{K}[T]^{-1} \right) \right) \subseteq \left(\downarrow [\![g]\!] \right) \setminus \left(\downarrow \left([\![g]\!] \setminus \mathcal{K}'[T]^{-1} \right) \right)$$

Proof. Straightforward consequence of Lemma 10

Definition 31. *We define the partial order $\sqsubseteq$ between environments as follows:*

$$\Gamma \sqsubseteq \Gamma' \iff \forall X \in \mathrm{dom}(\Gamma) : \Gamma(X) \subseteq \Gamma'(X)$$

Lemma 11. *For all Γ, Γ' such that $\Gamma \sqsubseteq \Gamma'$:*

$$\Gamma \vdash p : \mathcal{K} \implies \exists \mathcal{K}' : \ \Gamma' \vdash p : \mathcal{K}' \wedge \mathcal{K} \subseteq \mathcal{K}'$$

Proof. Suppose $\Gamma \sqsubseteq \Gamma'$ and $\Gamma \vdash p : \mathcal{K}$. By induction on the structure of p:

- $p = \mathbf{1}$: Trivial.
- $p = \sum_{i \in I} \mathtt{?a}_i\{g_i, T_i\} \,.\, p_i$: By kinding rule [T-+], must be:

  $$\mathcal{K} = \bigcup_{i \in I} \downarrow \left([\![g_i]\!] \cap \mathcal{K}_i[T_i]^{-1} \right), \text{ with } \Gamma \vdash p_i : \mathcal{K}_i$$

  By induction hypothesis:

  $$\Gamma' \vdash p_i : \mathcal{K}'_i, \text{ with } \mathcal{K}_i \subseteq \mathcal{K}'_i \text{ for all } i \in I$$

  The thesis follows by Item 1 of Corollary 1.
- $p = \bigoplus_{i \in I} \mathtt{!a}_i\{g_i, T_i\} \,.\, p_i$: Similar to the external choice case, using Item 2 of Corollary 1.
- $p = X$: Trivial by Definition 31 and kinding rule [T-Var].
- $p = \mathrm{rec}\, X \,.\, p'$: By kinding rule [T-Rec], must be:

  $$\frac{\exists \mathcal{K}, \mathcal{K}' : \ \Gamma\{\mathcal{K}/X\} \vdash p' : \mathcal{K}'}{\Gamma \vdash \mathrm{rec}\, X \,.\, p' : \bigcup \{ \mathcal{K} \mid \Gamma\{\mathcal{K}/X\} \vdash p' : \mathcal{K}' \wedge \mathcal{K} \subseteq \mathcal{K}' \} = \mathcal{K}}$$

  By the induction hypothesis:

  $$\frac{\exists \mathcal{K}, \mathcal{K}' : \ \Gamma'\{\mathcal{K}/X\} \vdash p' : \mathcal{K}'}{\Gamma' \vdash \mathrm{rec}\, X \,.\, p' : \bigcup \{ \mathcal{K} \mid \Gamma'\{\mathcal{K}/X\} \vdash p' : \mathcal{K}' \wedge \mathcal{K} \subseteq \mathcal{K}' \} = \mathcal{K}'}$$

  It remains to show $\mathcal{K} \subseteq \mathcal{K}'$. To see this, take some $\mathcal{K}_0, \mathcal{K}'_0$ such that:

  $$\Gamma\{\mathcal{K}_0/X\} \vdash p' : \mathcal{K}'_0 \wedge \mathcal{K}_0 \subseteq \mathcal{K}'_0$$

  If such $\mathcal{K}_0, \mathcal{K}'_0$ do not exist, then $\mathcal{K}$ is empty, and so the thesis holds vacuously. Otherwise, by the induction hypothesis:

  $$\Gamma'\{\mathcal{K}_0/X\} \vdash p' : \mathcal{K}'_1, \text{ with } \mathcal{K}'_0 \subseteq \mathcal{K}'_1$$

  By the transitivity of $\subseteq$ we have $\mathcal{K}_0 \subseteq \mathcal{K}'_1$, and hence $\mathcal{K}_0 \subseteq \mathcal{K}'$, from which follows the thesis.

Lemma 12.

$$\mathcal{K} \subseteq \mathcal{K}' \wedge \Gamma\{\mathcal{K}/x\} \vdash p : \tilde{\mathcal{K}} \implies \Gamma\{\mathcal{K}'/x\} \vdash p : \tilde{\mathcal{K}'} \wedge \tilde{\mathcal{K}} \subseteq \tilde{\mathcal{K}'}$$

Proof. Straightforward consequence of Lemma 11.

We recall the fundamental notion of clock region, first introduced in [2]. Our definition is more similar to the one appeared in [18].

Definition 32. *Let d be the largest constant occuring in the guards of p. A* p-region *is a set of clock valuations $\mathcal{K}$ such that:*

1. *There exists a guard g, with maximal constant bounded by d, such that $[\![g]\!] = \mathcal{K}$.*
2. *If $\mathcal{K}$ properly contains a p-region $\mathcal{K}'$, then $\mathcal{K}'$ is empty.*

In the following we will call p-regions simply regions, when p is clear from the context.

It is well-known (from [2]) that the set of regions is finite, and hence it is a complete lattice with, respectively, set union and intersection as join and meet. Furthermore, sets of regions are closed under the operators past and inverse reset ([18],[8]). Since all the kinding rules in Figure 2(except [T-Rec]) use only these operators together with finite union, intersection and difference, it follows that the kind of not recursive TSTs is a set of regions. By the Tarski-Knaster fixed point theorem (), together with Lemma 12, it follows that kinding rule [T-Rec] computes a greatest fixed point in the power set of $\mathbb{V}$. Again by the fixed point theorem, greatest fixed points of monotone functionals are greatest lower bounds of the descending chain of the functional starting from the top of the lattice, $\mathbb{V}$ in our case. Since $\mathbb{V}$ corresponds to the top of the power set of regions, we can conclude that the greatest fixed points is a set of regions as well, and can be computed in a finite number of steps.

Theorem 3. *Kind inference is decidable.*

Proof. Straigthforward by the above considerations.

The above observations, together with the fact that sets of regions can be represented by guards, provides us with an algorithm for constructing the canonical compliant for kindable TSTs. The idea is similar to that of duality in the untimed setting. Although uniqueness of kinding holds (see Lemma 13), there may be many different guards representing the same set of clock valuations; however, this is not an issue, as kind inference can always put guards in normal form.

Lemma 13. *For all $\Gamma, p, \mathcal{K}, \mathcal{K}'$:*

$$\Gamma \vdash p : \mathcal{K} \wedge \Gamma \vdash p : \mathcal{K}' \implies \mathcal{K} = \mathcal{K}'$$

Proof. Straightforward, by structural induction.

In the following all the substitutions are assumed to be capture avoiding.

Lemma 14 (Substitution). *Let $\Gamma \vdash p' : \mathcal{K}'$. Then, for all p:*

$$\Gamma\{^{\mathcal{K}'}/_X\} \vdash p : \mathcal{K} \iff \Gamma \vdash p\{^{p'}/_X\} : \mathcal{K}$$

Proof. We prove that, under the assumption $\Gamma \vdash p' : \mathcal{K}'$, it holds the following:

1. $\Gamma\{^{\mathcal{K}'}/_X\} \vdash p : \mathcal{K} \implies \exists \mathcal{K}'' : \ \Gamma \vdash p\{^{p'}/_X\} : \mathcal{K}'' \wedge \mathcal{K} \subseteq \mathcal{K}''$.
2. $\Gamma \vdash p\{^{p'}/_X\} : \mathcal{K} \implies \exists \mathcal{K}'' : \ \Gamma\{^{\mathcal{K}'}/_X\} \vdash p : \mathcal{K}'' \wedge \mathcal{K} \subseteq \mathcal{K}''$.

The thesis follows by their composition. To prove Item 1, assume $\Gamma\{^{\mathcal{K}'}/_X\} \vdash p : \mathcal{K}$. We proceed by induction on the typing rules.

- [T-1] Trivial.
- [T-$\oplus$]

$$\frac{\Gamma\{^{\mathcal{K}'}/_X\} \vdash p_i : \mathcal{K}_i}{\Gamma\{^{\mathcal{K}'}/_X\} \vdash \bigoplus \mathtt{!a}_i\{g_i, T_i\} \,.\, p_i : \left(\bigcup\!\downarrow [\![g_i]\!]\right) \setminus \left(\bigcup\!\downarrow ([\![g_i]\!] \setminus \mathcal{K}_i[T_i]^{-1})\right)}$$

  By the induction hypothesis:

$$\frac{\Gamma \vdash p_i\{^{p'}/_X\} : \tilde{\mathcal{K}}_i \supseteq \mathcal{K}_i}{\Gamma \vdash \bigoplus \mathtt{!a}_i\{g_i, T_i\} \,.\, (p_i\{^{p'}/_X\}) : \left(\bigcup\!\downarrow [\![g_i]\!]\right) \setminus \left(\bigcup\!\downarrow ([\![g_i]\!] \setminus \tilde{\mathcal{K}}_i[T_i]^{-1})\right)}$$

  The thesis follows by Item 1 of Corollary 1.
- [T-+] Similar to [T-$\oplus$], using Item 2 of Corollary 1.
- [T-Var] Trivial.
- [T-Rec] Clearly, p must match the following shape: $\mathrm{rec}\, Y \,.\, p''$. Assume $X \neq Y$ (otherwise the thesis is trivial); then:

$$\frac{\exists \mathcal{K}_0, \mathcal{K}'_0 : \ \Gamma\{^{\mathcal{K}'}/_X\}\{^{\mathcal{K}_0}/_Y\} \vdash p'' : \mathcal{K}'_0}{\Gamma\{^{\mathcal{K}'}/_X\} \vdash \mathrm{rec}\, Y \,.\, p'' : \bigcup \left\{ \tilde{\mathcal{K}} \mid \Gamma\{^{\mathcal{K}'}/_X\}\{^{\tilde{\mathcal{K}}}/_Y\} \vdash p'' : \tilde{\mathcal{K}}' \wedge \tilde{\mathcal{K}} \subseteq \tilde{\mathcal{K}}' \right\} = \mathcal{K}}$$

  Since $X \neq Y$:

$$p\{^{p'}/_X\} = \mathrm{rec}\, Y \,.\, (p''\{^{p'}/_X\})$$

  and, for all $\mathcal{K}_0, \mathcal{K}'_0$:

$$\Gamma\{^{\mathcal{K}'}/_X\}\{^{\mathcal{K}_0}/_Y\} \vdash p' : \mathcal{K}'_0 \iff \Gamma\{^{\mathcal{K}_0}/_Y\}\{^{\mathcal{K}'}/_X\} \vdash p' : \mathcal{K}'_0 \tag{9}$$

  The kinding derivation for $p\{^{p'}/_X\}$ must be as follows:

$$\frac{\exists \mathcal{K}_0, \mathcal{K}'_0 : \ \Gamma\{^{\mathcal{K}_0}/_Y\} \vdash (p''\{^{p'}/_X\}) : \mathcal{K}'_0}{\Gamma \vdash p\{^{p'}/_X\} : \bigcup \left\{ \tilde{\mathcal{K}} \mid \Gamma\{^{\tilde{\mathcal{K}}}/_Y\} \vdash (p''\{^{p'}/_X\}) : \tilde{\mathcal{K}}' \wedge \tilde{\mathcal{K}} \subseteq \tilde{\mathcal{K}}' \right\} = \mathcal{K}''} \tag{10}$$

  We first show that the premise of Equation (10) holds. Since substitutions are capture avoiding, Y is not free in p', and hence $\Gamma\{^{\mathcal{K}}/_Y\} \vdash p' : \mathcal{K}'$ as well as Γ. But then, by Equation (9) together with the induction hypothesis, it

follows the thesis. It remains to show $\mathcal{K} \subseteq \mathcal{K}''$. If $\mathcal{K}$ is empty the thesis holds trivially. Otherwise, take some $\tilde{\mathcal{K}}, \tilde{\mathcal{K}}'$ such that:

$$\Gamma\{\mathcal{K}'/X\}\{\tilde{\mathcal{K}}/Y\} \vdash p'' : \tilde{\mathcal{K}}' \wedge \tilde{\mathcal{K}} \subseteq \tilde{\mathcal{K}}'$$

By Equation (9), together with the induction hypothesis, we have that, for some $\tilde{\mathcal{K}}''$:

$$\Gamma\{\tilde{\mathcal{K}}/Y\} \vdash p''\{p'/X\} : \tilde{\mathcal{K}}'' \supseteq \tilde{\mathcal{K}}'$$

from which follows the thesis.

To prove Item 2, assume $\Gamma \vdash p\{p'/X\} : \mathcal{K}$. We proceed by induction on the structure of p.

- $p = \mathbf{1}$. Trivial.
- $p = \bigoplus \mathtt{!a}_i\{g_i, T_i\}\,.\,p_i$.

$$\frac{\Gamma \vdash p_i\{p'/X\} : \mathcal{K}_i}{\Gamma \vdash \bigoplus \mathtt{!a}_i\{g_i, T_i\}\,.\,(p_i\{p'/X\}) : \bigcup{\downarrow}\, [\![g_i]\!] \setminus \bigcup{\downarrow}\, ([\![g_i]\!] \setminus \mathcal{K}_i[T_i]^{-1})}\,[\text{T-}\oplus]$$

By induction hypothesis:

$$\frac{\Gamma\{\mathcal{K}_i/X\} \vdash p_i : \tilde{\mathcal{K}}_i \supseteq \mathcal{K}_i}{\Gamma\{\mathcal{K}'/X\} \vdash \bigoplus \mathtt{!a}_i\{g_i, T_i\}\,.\,p_i : \bigcup{\downarrow}\, [\![g_i]\!] \setminus \bigcup{\downarrow}\, ([\![g_i]\!] \setminus \tilde{\mathcal{K}}_i[T_i]^{-1})}\,[\text{T-}\oplus]$$

The thesis follows by Item 2 of Corollary 1.

- $p = \sum \mathtt{?a}_i\{g_i, T_i\}\,.\,p_i$. Similar to the internal choice case, using Item 1 of Corollary 1.
- $p = Y$. If $Y \neq X$ the thesis follows trivially. Otherwise, let $p = X$. Then $\Gamma \vdash p\{p'/X\} = p' : \mathcal{K}'$ and $\Gamma\{\mathcal{K}'/X\} \vdash p = X : \mathcal{K}'$.
- $p = \mathrm{rec}\, Y\,.\,p''$. Assume $X \neq Y$ (otherwise the thesis holds trivially).

$$\frac{\exists \mathcal{K}_0, \mathcal{K}'_0 :\ \Gamma\{\mathcal{K}_0/Y\} \vdash (p''\{p'/X\}) : \mathcal{K}'_0}{\Gamma \vdash p\{p'/X\} : \bigcup \left\{ \tilde{\mathcal{K}} \mid \Gamma\{\tilde{\mathcal{K}}/Y\} \vdash (p''\{p'/X\}) : \tilde{\mathcal{K}}' \wedge \tilde{\mathcal{K}} \subseteq \tilde{\mathcal{K}}' \right\} = \mathcal{K}}$$

As in the proof of Item 1, we have $\Gamma\{\mathcal{K}/Y\} \vdash p' : \mathcal{K}'$. Then, by Equation (9) and the induction hypothesis:

$$\frac{\exists \mathcal{K}_0, \mathcal{K}'_0 :\ \Gamma\{\mathcal{K}'/X\}\{\mathcal{K}_0/Y\} \vdash p'' : \mathcal{K}'_0}{\Gamma\{\mathcal{K}'/X\} \vdash \mathrm{rec}\, Y\,.\,p'' : \bigcup \left\{ \tilde{\mathcal{K}} \mid \Gamma\{\mathcal{K}'/X\}\{\tilde{\mathcal{K}}/Y\} \vdash p'' : \tilde{\mathcal{K}}' \wedge \tilde{\mathcal{K}} \subseteq \tilde{\mathcal{K}}' \right\} = \mathcal{K}''}$$

It remains to prove $\mathcal{K} \subseteq \mathcal{K}''$. If $\mathcal{K}$ is empty the thesis holds trivially. Otherwise, take some $\tilde{\mathcal{K}}, \tilde{\mathcal{K}}'$ such that:

$$\Gamma\{\tilde{\mathcal{K}}/Y\} \vdash p''\{p'/X\} : \tilde{\mathcal{K}}' \wedge \tilde{\mathcal{K}} \subseteq \tilde{\mathcal{K}}'$$

By Equation (9), together with the induction hypothesis, we have that, for some $\tilde{\mathcal{K}}''$:

$$\Gamma\{\mathcal{K}'/X\}\{\tilde{\mathcal{K}}/Y\} \vdash p'' : \tilde{\mathcal{K}}'' \supseteq \tilde{\mathcal{K}}'$$

from which follows the thesis.

Lemma 15. *Let $\Gamma \vdash p' : \mathcal{K}$, with X not free in p'. Then, for all p such that p is kindable with environment $\Gamma\{\mathcal{K}/X\}$:*

$$\mathrm{co}_{\Gamma\{\mathcal{K}/X\}}(p)\,\{\mathrm{co}_\Gamma(p')/X\} = \mathrm{co}_\Gamma(p\{p'/X\})$$

Proof. By induction on the structure of p:

- $p = \bigoplus \mathtt{!a}_i\{g_i, T_i\}\,.\,p_i$:

  $\mathrm{co}_{\Gamma\{\mathcal{K}/X\}}(p)\,\{\mathrm{co}_\Gamma(p')/X\} =$

  $= \sum \mathtt{?a}_i\{g_i, T_i\}\,.\,(\mathrm{co}_{\Gamma\{\mathcal{K}/X\}}(p_i)\,\{\mathrm{co}_\Gamma(p')/X\})$
  $= \sum \mathtt{?a}_i\{g_i, T_i\}\,.\,\mathrm{co}_\Gamma(p_i\{p'/X\})$ by induction hypothesis
  $= \mathrm{co}_\Gamma(p\{p'/X\})$

- $p = \sum \mathtt{?a}_i\{g_i, T_i\}\,.\,p_i$: Let $\Gamma\{\mathcal{K}/X\} \vdash p_i : \mathcal{K}_i$ for all $i \in I$. Then:

  $\mathrm{co}_{\Gamma\{\mathcal{K}/X\}}(p)\,\{\mathrm{co}_\Gamma(p')/X\} =$

  $= \bigoplus \mathtt{!a}_i\{g_i \wedge \mathcal{K}_i[T_i]^{-1}, T_i\}\,.\,(\mathrm{co}_{\Gamma\{\mathcal{K}/X\}}(p_i)\,\{\mathrm{co}_\Gamma(p')/X\})$
  $= \bigoplus \mathtt{!a}_i\{g_i \wedge \mathcal{K}_i[T_i]^{-1}, T_i\}\,.\,\mathrm{co}_\Gamma(p_i\{p'/X\})$ by induction hypothesis
  $= \mathrm{co}_\Gamma(p\{p'/X\})$ by lemma 14

- $p = \mathbf{1}$: Trivial
- $p = Y$: Trivial, whether or not $Y = X$
- $p = \mathrm{rec}\,Y\,.\,p''$: If $Y = X$ is trivial. Otherwise, assume $\Gamma\{\mathcal{K}/X\} \vdash \mathrm{rec}\,Y\,.\,p'' : \mathcal{K}'$. Then:

  $\mathrm{co}_{\Gamma\{\mathcal{K}/X\}}(p)\,\{\mathrm{co}_\Gamma(p')/X\} =$

  $= \mathrm{rec}\,Y\,.\,(\mathrm{co}_{\Gamma\{\mathcal{K}/X\}\{\mathcal{K}'/Y\}}(p'')\,\{\mathrm{co}_\Gamma(p')/X\})$
  $= \mathrm{rec}\,Y\,.\,(\mathrm{co}_{\Gamma\{\mathcal{K}'/Y\}\{\mathcal{K}/X\}}(p'')\,\{\mathrm{co}_\Gamma(p')/X\})$ since $X \neq Y$
  $= \mathrm{rec}\,Y\,.\,(\mathrm{co}_{\Gamma\{\mathcal{K}'/Y\}\{\mathcal{K}/X\}}(p'')\,\{\mathrm{co}_{\Gamma\{\mathcal{K}'/Y\}}(p')/X\})$ since substitutions are capture avoiding
  $= \mathrm{rec}\,Y\,.\,\mathrm{co}_{\Gamma\{\mathcal{K}'/Y\}}(p''\{p'/X\})$ by induction hypothesis
  $= \mathrm{co}_{\Gamma\{\mathcal{K}'/Y\}}(p\{p'/X\})$
  $= \mathrm{co}_\Gamma(p\{p'/X\})$ since Y is not free in $p\{p'/X\}$

Lemma 16. *For all TSTs p and q, we have that: $p \equiv q \implies \mathrm{co}(p) \equiv \mathrm{co}(q)$.*

Proof. Straightforward consequence of Lemma 15. □

Lemma 17. *Every closed p is structural equivalent to a TST with one of the following shapes:*

1. $\mathbf{1}$
2. $\bigoplus \mathtt{!a}_i\{g_i, T_i\}\,.\,p_i$
3. $\sum \mathtt{?a}_i\{g_i, T_i\}\,.\,p_i$

Proof. Trivial. □

Theorem 4. *If $\vdash p : \mathcal{K}$ and $\nu \in \mathcal{K}$, then $(p, \nu) \bowtie (\mathrm{co}(p)\,, \nu)$.*

Proof. By Lemma 17 every TST can be unfolded to a sum or $\mathbf{1}$ and its dual too (lemma 16). Then, let:

$$\mathcal{R} = \{((p,\nu),(\mathrm{co}(p),\nu)) \mid \exists \mathcal{K} : \ \Gamma \vdash p : \mathcal{K} \wedge \nu \in \mathcal{K}\}$$

We proceed by showing that $\mathcal{R}$ is a coinductive compliance relation (Definition 7). Assume $\Gamma \vdash p : \mathcal{K} \wedge \nu \in \mathcal{K}$. By cases on p:

- $p = \mathbf{1}$: Trivial
- $p = \bigoplus \mathtt{!a}_i\{g_i, T_i\}\,.\,p_i$:

  $$\mathrm{co}(p) = \sum \mathtt{?a}_i\{g_i, T_i\}\,.\,\mathrm{co}(p_i)$$

  By the kinding rule [T-⊕] follows trivially $\nu \in \mathtt{rdy}(p)$. It remains to show:

  $$\forall \delta, i : \ \nu + \delta \in [\![g_i]\!] \implies (p_i, (\nu+\delta)[T_i])\ \mathcal{R}\ (\mathrm{co}(p_i), (\nu+\delta)[T_i])$$

  Let $\vdash p_i : \mathcal{K}_i$. By the typing rule [T-⊕] we have that:

  $$\begin{aligned} \mathcal{K} = \\ \left(\textstyle\bigcup_{i\in I} {\downarrow}\,[\![g_i]\!]\right) \setminus \left(\textstyle\bigcup_{i\in I} {\downarrow}\,([\![g_i]\!] \setminus \mathcal{K}_i[T_i]^{-1})\right) = \\ \left(\textstyle\bigcup_{i\in I} {\downarrow}\,[\![g_i]\!]\right) \setminus \{\nu \mid \exists \delta, i : \nu + \delta \in [\![g_i]\!] \wedge \nu + \delta[T_i] \notin \mathcal{K}_i\} \end{aligned}$$

  from which follows the thesis.
- $p = \sum \mathtt{?a}_i\{g_i, T_i\}\,.\,p_i$:

  $$\mathrm{co}(p) = \bigoplus \mathtt{!a}_i\{g_i \wedge \mathcal{K}_i, T_i\}\,.\,\mathrm{co}(p_i)$$

  with $\vdash p_i : \mathcal{K}_i$. By the typing rule [T-+]:

  $$\mathcal{K} = \bigcup {\downarrow}\left([\![g_i]\!] \cap \mathcal{K}_i[T_i]^{-1}\right) = \{\nu \mid \exists \delta, i : \ \nu + \delta \in [\![g_i]\!] \wedge (\nu+\delta)[T_i] \in \mathcal{K}_i\}$$

  from which follows the thesis. □

Definition 33. *We define the function Φ from TSTs to sets of clock valuations as follows:*

$$\Phi(p) \stackrel{def}{=} \{\nu \mid \exists q, \eta : (p,\nu) \bowtie (q,\eta)\}$$

Lemma 18. *Let $\Gamma = X_1 : p_1, \ldots, X_m : p_m$, for some vectors $\boldsymbol{X}, \boldsymbol{p}$ of length m. Then, for all p such that $\mathtt{fv}(p) \subseteq \boldsymbol{X}$:*

$$\Gamma \vdash p : \mathcal{K} \implies \Phi(p\{\boldsymbol{p}/\boldsymbol{X}\}) \subseteq \mathcal{K}$$

Proof. We start with an auxiliar definition. The recursion annidation level (RL) for a TST is inductively defined as follows:

$$\begin{aligned} \mathrm{RL}(\mathbf{1}) &\stackrel{\text{def}}{=} 0 \\ \mathrm{RL}(\textstyle\sum_{i\in I} \mathtt{?a}_i\{g_i, T_i\}\,.\,p_i) &\stackrel{\text{def}}{=} max(\{\mathrm{RL}(p_i)\}_{i\in I}) \\ \mathrm{RL}(\textstyle\bigoplus_{i\in I} \mathtt{!a}_i\{g_i, T_i\}\,.\,p_i) &\stackrel{\text{def}}{=} max(\{\mathrm{RL}(p_i)\}_{i\in I}) \\ \mathrm{RL}(X) &\stackrel{\text{def}}{=} 0 \\ \mathrm{RL}(\mathrm{rec}\, X\,.\,p) &\stackrel{\text{def}}{=} 1 + \mathrm{RL}(p) \end{aligned}$$

We then define the partial order $\tilde{\sqsubseteq}$ as:

$$p \,\tilde{\sqsubseteq}\, q \iff \mathrm{RL}(p) < \mathrm{RL}(q) \vee (\mathrm{RL}(p) = \mathrm{RL}(q) \wedge p \leq q)$$

where $p \leq q$ iff p is syntactically lower than q. It is trivial to check that $\tilde{\sqsubseteq}$ is a well-founded ordering. We then proceed by well-founded induction on $\tilde{\sqsubseteq}$. We have the following cases, according to the form of p:

- $p = \mathbf{1}$. Since $\mathcal{K} = \mathbb{V}$ (by kinding rule [T-1]), the thesis follows trivially.
- $p = X_i$, for some $i \in \{1, \ldots, m\}$. $\mathcal{K} = \Gamma(X_i) = \Phi(p_i) = \Phi(X_i\{\boldsymbol{p}/\boldsymbol{X}\})$.
- $p = \sum_{i \in I} \mathtt{?a}_i\{g_i, T_i\} \,.\, p_i$.

$$\frac{\Gamma \vdash p_i : \mathcal{K}_i \qquad \text{for } i \in I}{\Gamma \vdash \sum_{i \in I} \mathtt{?a}_i\{g_i, T_i\} \,.\, p_i : \bigcup_{i \in I} \downarrow \left([\![g_i]\!] \cap \mathcal{K}_i[T_i]^{-1} \right) = \mathcal{K}} \text{ [T-+]}$$

Since, for all $i \in I$ it holds $p_i \,\tilde{\sqsubseteq}\, p$, by the induction hypothesis:

$$\Phi(p_i\{\boldsymbol{p}/\boldsymbol{X}\}) \subseteq \mathcal{K}_i \tag{11}$$

Now suppose, by contradiction, $\Phi(p\{\boldsymbol{p}/\boldsymbol{X}\}) \not\subseteq \mathcal{K}$. Then there exist ν, η, q such that $\Phi(p\{\boldsymbol{p}/\boldsymbol{X}\}) \ni \nu \notin \mathcal{K}$ and $(p\{\boldsymbol{p}/\boldsymbol{X}\}, \nu) \bowtie (q, \eta)$. By Definition 7 we have $q = \bigoplus_{j \in J} \mathtt{!a}_j\{g_j, T_j\} \,.\, q_j$, with $\eta \in \mathtt{rdy}(q)$ and

$$\forall \delta, j : \ \eta + \delta \in [\![g_j]\!] \implies$$

$$\exists i : \mathtt{a}_i = \mathtt{a}_j \wedge \nu + \delta \in [\![g_i]\!] \wedge (p_i\{\boldsymbol{p}/\boldsymbol{X}\}, (\nu + \delta)[R_i]) \bowtie (q_j, (\eta + \delta)[R_j])$$

But then, by Definition 33 and Equation (11):

$$(\nu + \delta)[R_i] \in \Phi(p_i\{\boldsymbol{p}/\boldsymbol{X}\}) \subseteq \mathcal{K}_i$$

Since $\mathcal{K}$ is past closed (i.e. for all ν, δ: $\nu + \delta \in \mathcal{K} \implies \nu \in \mathcal{K}$), it follows $\nu \in \mathcal{K}$, a contradiction.

- $p = \bigoplus_{i \in I} \mathtt{!a}_i\{g_i, T_i\} \,.\, p_i$. Similar to the external choice case.
- $p = \mathrm{rec}\, X \,.\, p'$.

$$\frac{\exists \mathcal{K}_0, \mathcal{K}'_0 : \ \Gamma\{\mathcal{K}_0/X\} \vdash p' : \mathcal{K}'_0}{\Gamma \vdash \mathrm{rec}\, X \,.\, p' : \bigcup \{\mathcal{K} \mid \exists \mathcal{K}' : \ \Gamma\{\mathcal{K}/X\} \vdash p' : \mathcal{K}' \wedge \mathcal{K} \subseteq \mathcal{K}'\} = \mathcal{K}} \text{ [T-Rec]}$$

It is enough to show:

$$\Gamma\{\Phi'/X\} \vdash p' : \mathcal{K}' \wedge \Phi' \subseteq \mathcal{K}'$$

where $\Phi' = \Phi(p'\{\boldsymbol{p}'/\boldsymbol{X}'\})$ and:

$\boldsymbol{X}' = (\boldsymbol{X}, X)$	if $X \notin \boldsymbol{X}$
$\boldsymbol{X}' = \boldsymbol{X}$	otherwise
$\boldsymbol{p}' = (p_1, \ldots, p_{i-1}, p\{\boldsymbol{p}/\boldsymbol{X}\}, p_{i+1}, \ldots, p_m)$	if there exists i such that $X = X_i$
$\boldsymbol{p}' = (\boldsymbol{p}, p\{\boldsymbol{p}/\boldsymbol{X}\})$	otherwise

Since $\mathrm{RL}(p') < \mathrm{RL}(p)$, the thesis follows by induction hypothesis.

□

Theorem 5. *If* $\vdash p : \mathcal{K}$ *and* $\exists q, \eta.\ (p, \nu) \bowtie (q, \eta)$*, then* $\nu \in \mathcal{K}$*.*

Proof. Straightforward consequence of Lemma 18. □

Lemma 19. *For all* p, q, ν, η *and* p', ν' *such that* $\vdash p' : \mathcal{K}$ *and* $\nu' \in \mathcal{K}$*:*

$$(p, \nu) \bowtie (p', \nu') \wedge (\mathrm{co}(p'), \nu') \bowtie (q, \eta) \implies (p, \nu) \bowtie (q, \eta)$$

Proof. Let:

$$\mathcal{R} = \{((p, \nu), (q, \eta)) \mid \exists p', \nu' : (p, \nu) \bowtie (p', \nu') \wedge (\mathrm{co}(p'), \nu') \bowtie (q, \eta)\}$$

We prove $\mathcal{R}$ is a coinductive compliance relation (Definition 7). By cases on the shape of p:

- $p = \mathbf{1}$: trivial.
- $p = \bigoplus \mathtt{!a}_i\{g_i, T_i\}.\, p_i$: it must be:

  $$p' = \sum \mathtt{?a}_j\{g_j, T_j\}.\, {p_j}'$$

  and

  $$\mathrm{co}(p') = \bigoplus \mathtt{!a}_j\{g_j \wedge \mathcal{K}_j[T_j]^{-1}, T_j\}.\, \mathrm{co}({p_j}')$$

  with $\vdash p_j : \mathcal{K}_j$ for all $j \in J$ and, by Definition 7:
  $\forall \delta, i :\ \nu + \delta \in [\![g_i]\!] \implies \exists j :\ \mathtt{a}_i = \mathtt{a}_j \ \wedge\ \nu' + \delta \in [\![g_j]\!] \ \wedge$
  $(p_i, (\nu + \delta)[T_i]) \bowtie ({p_j}', (\nu' + \delta)[T_j]) \ \wedge\ \nu + \delta \in \mathtt{rdy}(p)$.
  Hence:

  $$q = \sum \mathtt{?a}_k\{g_k, T_k\}.\, p_k$$

  with:
  $\forall \delta, j :\ \nu' + \delta \in [\![g_j]\!] \implies \exists k :\ \mathtt{a}_k = \mathtt{a}_j \ \wedge\ \eta + \delta \in [\![g_k]\!] \ \wedge$
  $(\mathrm{co}({p_j}'), (\nu' + \delta)[T_j]) \bowtie (q_k, (\eta + \delta)[T_k]) \ \wedge\ \nu' \in \mathtt{rdy}(\mathrm{co}(p'))$.
  Now, assume $\nu + \delta \in g_i$ for some δ, i, and suppose, by contradiction, $\nu' + \delta \notin \mathcal{K}_j[T_j]^{-1}$ for some j such that $\mathtt{a}_i = \mathtt{a}_j$. But then should be $(p_i, (\nu + \delta)[T_i]) \bowtie ({p_j}', (\nu' + \delta)[T_j])$ and $(\nu' + \delta)[T_j] \notin \mathcal{K}_j$, absurd by Theorem 5.
- $p = \sum \mathtt{?a}_i\{g_i, T_i\}.\, p_i$: it must be:

  $$p' = \bigoplus \mathtt{!a}_j\{g_j, T_j\}.\, {p_j}'$$

  and

  $$\mathrm{co}(p') = \sum \mathtt{?a}_j\{g_j, T_j\}.\, \mathrm{co}(p_j)$$

  with:
  $\forall \delta, j :\ \nu' + \delta \in [\![g_j]\!] \implies \exists i :\ \mathtt{a}_j = \mathtt{a}_i \ \wedge\ \nu + \delta \in [\![g_i]\!] \ \wedge$

$(p_j{}', \nu' + \delta[T_j]) \bowtie (p_i, \nu + \delta[T_j]) \ \wedge \ \nu' \in \mathtt{rdy}(p')$.
Hence:

$$q = \bigoplus \mathtt{?a}_k\{g_k, T_k\} \,.\, p_k$$

with:
$\forall \delta, k : \ \eta + \delta \in [\![g_k]\!] \implies \exists j : \ \mathtt{a}_k = \mathtt{a}_j \ \wedge \ \nu' + \delta \in [\![g_j]\!] \ \wedge$
$(q_k, (\eta + \delta)[T_k]) \bowtie (p_j{}', (\nu + \delta)[T_i]) \ \wedge \ \eta \in \mathtt{rdy}(q)$. By the composition of the above follows the thesis.

Theorem 6. *If $p \bowtie p'$ and $\mathrm{co}(p') \bowtie q$, then $p \bowtie q$.*

Proof. Straightforward after Lemma 19. □

Theorem 7. *For all TSTs p, q: $q \bowtie p \implies q \sqsubseteq \mathrm{co}(p)$*

Proof. We extend definition 10 as: $(p, \nu)^{\bowtie} \stackrel{\text{def}}{=} \{(q, \eta) \mid (p, \nu) \bowtie (q, \eta)\}$, and:

$$(p, \nu) \sqsubseteq (q, \eta) \qquad \text{iff} \qquad (p, \nu)^{\bowtie} \supseteq (q, \eta)^{\bowtie}$$

We then prove the more general fact:
For all p, q such that $\vdash q : \mathcal{K}$ and $\eta \in \mathcal{K}$:

$$(q, \eta) \bowtie (p, \nu) \implies (p, \nu) \sqsubseteq (\mathrm{co}(q), \eta)$$

Then, let $\vdash q : \mathcal{K}$ and $\eta \in \mathcal{K}$, and suppose $(q, \eta) \bowtie (p, \nu)$. We have to show $(p, \nu) \sqsubseteq (\mathrm{co}(q), \eta)$. So, assume $(\mathrm{co}(q), \eta) \bowtie (p', \nu')$. By Lemma 19: $(p, \nu) \bowtie (p', \nu')$.

Theorem 2. *For all closed p, there exists some $\mathcal{K}$ such that $\vdash p : \mathcal{K}$.*

Proof. We prove the more general fact: For all Γ, p such that $\mathtt{fv}(p) \subseteq \mathrm{dom}(\Gamma)$ there exists some $\mathcal{K}$ such that $\Gamma \vdash p : \mathcal{K}$. By induction on the structure of p.

$p = \mathbf{1}$: Trivial.
$p = \sum_{i \in I} \mathtt{?a}_i\{g_i, T_i\} \,.\, p_i$: By induction hypothesis, for every $i \in I$ there exists some $\mathcal{K}_i$ such that $\Gamma \vdash p_i : \mathcal{K}_i$, and hence we can construct $\mathcal{K}$ according to rule [T-+].
$p = \bigoplus_{i \in I} \mathtt{!a}_i\{g_i, T_i\} \,.\, p_i$: Similar to the external choice case.
$p = X$: Since $\mathtt{fv}(p) \subseteq \mathrm{dom}(\Gamma)$, by rule [T-Var] we have $\mathcal{K} = \Gamma(X)$.
$p = \mathrm{rec}\, X \,.\, p'$: Since $\mathtt{fv}(p) \subseteq \mathrm{dom}(\Gamma)$, for every $\mathcal{K}'$:

$$\mathtt{fv}(p') = \mathtt{fv}(p) \cup \{X\} \subseteq \mathrm{dom}(\Gamma) \cup \{X\} = \mathrm{dom}(\Gamma\{\mathcal{K}'/X\})$$

By induction hypothesis:

$$\exists \mathcal{K}'' : \Gamma\{\mathcal{K}'/X\} \vdash p : \mathcal{K}''$$

By rule [T-Rec] toghether with the fixed point theorem follows the thesis.

Theorem 8. *If q admits a compliant, then: $p \sqsubseteq q \iff p \bowtie \mathrm{co}(q)$.*

Proof. For the ($\Rightarrow$) direction, assume that $p \sqsubseteq q$. Since q admits a compliant, by Theorem 5 there exists some $\mathcal{K}$ such that $\vdash q : \mathcal{K} \ni \eta_0$. By Theorem 4, it follows that $\mathrm{co}(q) \bowtie q$. Then, by Definition 10 we conclude that $p \bowtie \mathrm{co}(q)$.
For the ($\Leftarrow$) direction, assume that $p \bowtie \mathrm{co}(q)$, and let q' be such that $q' \bowtie q$. Then, by Theorem 6 we conclude that $q' \bowtie p$, from which the thesis follows. □

D Proofs for Section 4

To prove the main result of this appendix (Theorem 9) we generalise definition of compliance, by parameterising it on the LTS $\rightarrow$, parallel composition operator C$\circ$, and success states S. The set Z of the states of the LTS $\rightarrow$ contains terms of the form $P \circ Q$. Further, the notion of compliance we consider is *asymmetric*, meaning that if a client contract P is compliant with a server contract Q then, whenever a computation of $P \mid Q$ becomes stuck, the client has reached the success state [11].

Definition 34 (Generalised compliance). *Let T be a set of terms, and let $(Z, \circ, \mathtt{A}, \rightarrow, S)$ be an LTS, where $Z \subseteq \{P \circ Q \mid P, Q \in T\}$ and $S \subseteq Z$. We say that P is* compliant *with Q (written $P \multimap Q$) whenever:*

$$P \circ Q \;\rightarrow^*\; P' \circ Q' \;\not\rightarrow \qquad \textit{implies} \qquad P' \circ Q' \in S$$

Note that the notion of compliance $\bowtie$ introduced in Definition 6 can be seen as $\lhd \cap \rhd$, where the asymmetric compliance relations $\rhd$ and $\rhd$ are instances generalised compliance in Definition 34. In particular, $\lhd$ (the client-aware version) is obtained by instantiating $\rightarrow$ with the relation in Figure 1, $\circ$ with parallel $|$, and S with the set of terms of the form $(\mathbf{1}, \nu) \mid (q, \eta)$. The server-aware relation $\rhd$ is similar, with $\mathbf{1}$ in the right component.

We now formalise compliance for the monitoring semantics.

Definition 35 (Monitoring Compliance). *We say that a monitoring configuration γ is* deadlock *whenever (i) it is not the case that both p and q in γ are $\mathbf{1}$, and (ii) there is no λ such that $\gamma \xrightarrow{\lambda}\!\!\!\!\rightarrow$. We then write $(p, c, \nu) \bowtie_M (q, d, \eta)$ whenever:*

$$(p, c, \nu) \parallel (q, d, \eta) \twoheadrightarrow^* \gamma \qquad \textit{implies} \qquad \gamma \textit{ not deadlock}$$

We say that p and q are monitoring compliant *whenever $(p, [], \nu_0) \bowtie_M (q, [], \eta_0)$ (in short, $p \bowtie_M q$).*

Also monitoring compliance $\bowtie_M$ can be obtained as $\lhd_M \cap \rhd_M$, where $\lhd_M$ and $\rhd_M$ are instances of Definition 34. In particular, $\lhd_M$ is obtained by instantiating $\rightarrow$ in Definition 34 with the relation $\twoheadrightarrow$ in Figure 4, $\circ$ with $\parallel$, and S with the set of terms of the form $(\mathbf{1}, [], \nu) \parallel (q, [], \eta)$.

To show compliance and monitoring compliance equivalent, we introduce a notion of simulation (called *turn-simulation*) which is suitable to relate the monitoring semantics in Figure 4 with the one in Figure 1. This relation is between states of two LTSs $\rightarrow_1$ and $\rightarrow_2$, and it is parameterised over two sets S_1 and S_2 of success states. A state $(s_2, \rightarrow_2)$ turn-simulates $(s_1, \rightarrow_1)$ whenever each move of s_1 can be matched by a sequence of moves of s_2 (ignoring the labels), and stuckness of s_1 implies that s_2 will get stuck in at most one step. Further, turn-simulation must preserve success.

Definition 36 (Turn-simulation). *For $i \in \{1,2\}$, let $\rightarrow_i$ be an LTS over a state space Z_i, and let S_i be a set of states of $\rightarrow_i$. We say that a relation $\mathcal{R} \subseteq Z_1 \times Z_2$ is a* turn-simulation *iff $s_1 \mathcal{R} s_2$ implies:*

(a) $s_1 \rightarrow_1 s_1' \implies \exists s_2' : s_2 \rightarrow_2^* s_2'$ *and* $s_1' \mathcal{R} s_2'$
(b) $s_2 \rightarrow_2 s_2' \implies s_1 \rightarrow_1$ *or* $(s_1 \mathcal{R} s_2'$ *and* $s_2' \not\rightarrow_2)$
(c) $s_2 \in S_2 \implies s_1 \in S_1$

If there is a turn-simulation between s_1 and s_2 (written $s_1 \mathcal{R} s_2$), we say that s_2 turn-simulates s_1. We denote with $\preccurlyeq$ the greatest turn-simulation.

We say that $\mathcal{R}$ is a turn-bisimulation *iff both $\mathcal{R} \subseteq Z_1 \times Z_2$ and $\mathcal{R}^{-1} \subseteq Z_2 \times Z_1$ are turn-simulations.*

The following lemma relates turn-based simulation and compliance in two arbitrary LTSs. Whenever P and Q can be composed in parallel in both LTSs, and these compositions are turn-similar, then compliance can be transferred from one LTS to the other (in the other direction w.r.t. the simulation).

Lemma 20. *If $P \circ_1 Q \preccurlyeq P \circ_2 Q$ and $P \multimap_2 Q$, then $P \multimap_1 Q$.*

Proof. By Definition 34, assume that:

$$P \circ_1 Q \ \rightarrow_1^* \ P_1' \circ_1 Q_1' \ \not\rightarrow_1$$

Since $P \circ_1 Q \preccurlyeq P \circ_2 Q$ and $P \circ_1 Q \ \rightarrow_1^* \ P_1' \circ_1 Q_1'$, by item a of Definition 36 we have that there exist P_2', Q_2' such that $P \circ_2 Q \ \rightarrow_2^* \ P_2' \circ_1 Q_2'$ and $P_1' \circ_1 Q_1' \preccurlyeq P_2' \circ_2 Q_2'$. Now we have the following two cases:

- $P_2' \circ_2 Q_2' \not\rightarrow_2$. Since $P \multimap_2 Q$, by Definition 6 it must be $P_2' \circ_2 Q_2' \in S_2$. By item c of Definition 36 it follows that $P_1' \circ_1 Q_1' \in S_1$, from which we conclude that $P \multimap_1 Q$.
- $P_2' \circ_2 Q_2' \rightarrow_2 P_2'' \circ_2 Q_2''$. By item b of Definition 36 we have one of the following two cases:
  - $P_1' \circ_1 Q_1' \rightarrow_1$. This is not possible, because we have assumed that $P_1' \circ_1 Q_1'$ is stuck.
  - $P_1' \circ_1 Q_1' \preccurlyeq P_2'' \circ_2 Q_2''$ and $P_2'' \circ_2 Q_2'' \not\rightarrow_2$. Since $P_2'' \circ_2 Q_2''$ is stuck and $P \multimap_2 Q$, by Definition 6 it follows that $P_2'' \circ_2 Q_2'' \in S_2$. Hence, from $P_1' \circ_1 Q_1' \preccurlyeq P_2'' \circ_2 Q_2''$ and item c of Definition 36 it must be $P_1' \circ_1 Q_1' \in S_1$, from which the thesis $P \multimap_1 Q$ follows. □

The following lemma establishes that the two semantics of session types (Definitions 5 and 11) are turn-bisimilar.

Lemma 21. *$(p, \nu) \mid (q, \eta)$ is turn-bisimilar to $(p, [], \nu) \parallel (q, [], \eta)$.*

Proof. Let us consider the LTS defined by the relation $\rightarrow$ of Figure 1 and set of success states as $S_1 = \{(\mathbf{1}, \nu) \mid (q, \eta) \mid q \text{ TST}, \nu, \eta \in \mathbb{V}\}$. Also, consider the

LTS defined by the relation $\twoheadrightarrow_M$ of Figure 4 and set of success states as $S_2 = \{(\mathbf{1}, [], \nu) \parallel (q, [], \eta) \mid q \text{ TST}, \nu, \eta \in \mathbb{V}\}$. Let $\mathcal{R}$ be a relation defined as follows:

$$
\begin{aligned}
\mathcal{R} &= \mathcal{R}_1 \cup \mathcal{R}_2 \cup \mathcal{R}_3 \cup \mathcal{R}_{2B} \cup \mathcal{R}_{3B} \\
\mathcal{R}_1 &= \{((p,\nu) \mid (q,\eta),\ \ (p,[],\nu) \parallel (q,[],\eta)) \mid p, q \text{ TST}, \nu, \eta \in \mathbb{V}\} \\
\mathcal{R}_2 &= \{(([\mathtt{!a}\{g,R\}]\, p, \nu) \mid (q,\eta),\ \ (p,[\mathtt{!a}],\nu[R]) \parallel (q,[],\eta)) \mid (tsbP, q \text{ TST}, \nu \in [\![g]\!]\} \\
\mathcal{R}_3 &= \{(([\mathtt{!a}\{g,R\}]\, p, \nu) \mid ([\mathtt{!b}\{f,S\}]\, q, \eta),\ \ (p,[\mathtt{!a}],\nu[R]) \parallel (\mathtt{!b}\{f,S\}.q \oplus q', [], \eta)) \mid \cdots\} \\
\mathcal{R}_{2S} &= \{((p,\nu) \mid ([\mathtt{!a}\{f,S\}]\, q, \eta),\ \ (p,[],\nu) \parallel (q,[\mathtt{!a}],\nu[S])) \mid p, q \text{ TST}, \eta \in [\![f]\!]\} \\
\mathcal{R}_{3S} &= \{(([\mathtt{!a}\{g,R\}]\, p, \nu) \mid ([\mathtt{!b}\{f,S\}]\, q, \eta),\ \ (\mathtt{!a}\{g,R\}.p \oplus p', [], \nu) \parallel (q,[\mathtt{!b}],\eta[R]) \mid \cdots\}
\end{aligned}
$$

Let $s_1 = (p,\nu) \mid (q,\eta)$, and let $s_2 = (p,[],\nu) \parallel (q,[],\eta)$. Clearly, $s_1\ \mathcal{R}\ s_2$, hence to obtain the thesis we will prove that $\mathcal{R}$ is a turn-bisimulation. The proof is organised as follows. In **Part A** we will show that s_2 turn-simulates s_1 via $\mathcal{R}$, and in **Part B** the *vice-versa*, i.e. that s_1 turn-simulates s_2 via $\mathcal{R}$. Within each part, we proceed by cases on the form of s_1 and s_2: in **Case 1** we assume that $(s_1, s_2) \in \mathcal{R}_1$, in **Case 2** that $(s_1, s_2) \in \mathcal{R}_2$, and in **Case 3** that $(s_1, s_2) \in \mathcal{R}_3$. We omit cases for $\mathcal{R}_{2S}$ and $\mathcal{R}_{3S}$, since they are specular to cases $\mathcal{R}_2$ and $\mathcal{R}_3$. For each case, we show that items (a), (b), and (c) of Definition 36 hold. We will only consider the moves of the LHS of a composition $P \circ Q$; all the symmetric cases will be omitted.

Part A: s_2 turn-simulates s_1 via $\mathcal{R}$.

Case 1: Let $s_1 = (p,\nu) \mid (q,\eta)$ and $s_2 = (p,[],\nu) \parallel (q,[],\eta)$.

(a) To prove item (a) of Definition 36, we consider the possible moves of s_1:

- [S-$\oplus$]. We have:

$$\frac{(p,\nu) \xrightarrow{\tau} ([\mathtt{!a}\{g,R\}]\, p', \nu)}{s_1 \xrightarrow{\tau} ([\mathtt{!a}\{g,R\}]\, p', \nu) \mid (q,\eta) = s_1'}$$

  where the premise requires $p = \mathtt{!a}\{g,R\}.p' \oplus p''$ and $\nu \in [\![g]\!]$. Hence, by rule [M-$\oplus$] we have:

$$s_2 \xrightarrow{\mathsf{A}:\mathtt{!a}} (p', [\mathtt{!a}], \nu[R]) \parallel (q,[],\eta) = s_2'$$

  Then, $(s_1', s_2') \in \mathcal{R}_2 \subseteq \mathcal{R}$.
- [S-τ]. This case does not apply.
- [S-Del]. We have:

$$\frac{(p,\nu) \xrightarrow{\delta} (p', \nu+\delta) \qquad (q,\eta) \xrightarrow{\delta} (q', \eta+\delta)}{s_1 \xrightarrow{\delta} (p', \nu+\delta) \mid (q', \eta+\delta) = s_1'}$$

  The only rule which can be used in the premises of the above is [Del], which implies that $p = p'$, $q = q'$, $\nu + \delta \in \mathtt{rdy}(p)$ and $\eta + \delta \in \mathtt{rdy}(q)$. Then, the thesis follows by rule [M-Del].

(b) To prove item (b) of Definition 36, we consider the possible moves of s_2:

- [M-⊕]. We have $p = \texttt{!a}\{g, R\}.p' \oplus p''$, $\nu \in [\![g]\!]$, and:

$$s_2 \xrightarrow{\mathsf{A}:\texttt{!a}} (p', [\texttt{!a}], \nu[R]) \parallel (q, [], \eta) = s_2'$$

So, by rules [⊕] and [S-⊕] we have:

$$\frac{p \xrightarrow{\tau} ([\texttt{!a}\{g, R\}]\, p', \nu)}{s_1 \xrightarrow{\tau} ([\texttt{!a}\{g, R\}]\, p', \nu) \mid (q, \eta)} = s_1'$$

and we conclude that $(s_1', s_2') \in \mathcal{R}_2 \subseteq \mathcal{R}$.
- [M-+]. This case does not apply, since both buffers are empty.
- [M-Del]. We have $\nu + \delta \in \texttt{rdy}(p)$, $\eta + \delta \in \texttt{rdy}(q)$, and:

$$s_2 \xrightarrow{\delta} (p, [], \nu + \delta) \parallel (q, [], \eta + \delta) = s_2'$$

Then, rule [Del] yields $(p, \nu) \xrightarrow{\delta} (p, \nu + \delta)$ and $(q, \eta) \xrightarrow{\delta} (p, \eta + \delta)$. Hence, by rule [S-Del] we conclude that:

$$s_1 \xrightarrow{\delta} (p, \nu + \delta) \mid (q, \eta + \delta) = s_1'$$

and the thesis follows because $(s_1', s_2') \in \mathcal{R}_1 \subseteq \mathcal{R}$.

(c) To prove item (c) of Definition 36, assume that $s_2 \in S_2$. By definition of S_2, s_2 has the form $(\mathbf{1}, [], \nu) \parallel (q, [], \eta)$. Then, $s_1 = (\mathbf{1}, \nu) \mid (q, \eta) \in S_1$.

Case 2: Let $s_1 = ([\texttt{!a}\{g, R\}]\, p, \nu) \mid (q, \eta)$ and $s_2 = (p, [\texttt{!a}], R[\nu]) \parallel (q, [], \eta)$ with $\nu \in [\![g]\!]$.

(a) To prove item (a) of Definition 36, we consider the possible moves of s_1:
- [S-⊕]. We have:

$$\frac{(q, \eta) \xrightarrow{\tau} ([\texttt{!b}\{f, S\}]\, q', S[\eta])}{s_1 \xrightarrow{\tau} ([\texttt{!a}\{g, R\}]\, p, \nu) \mid ([\texttt{!b}\{f, S\}]\, q', S[\eta]) = s_1'}$$

where the premise requires $q = \texttt{!a}\{f, S\}.q' \oplus q''$ and $\eta \in [\![f]\!]$. Hence, by rule [M-⊕] we have:

$$s_2 \xrightarrow{\mathsf{B}:\texttt{!b}} (p', b, \nu) \parallel (q', [\texttt{!b}], \eta[S]) = s_2'$$

Then, $(s_1', s_2') \in \mathcal{R}_2 \subseteq \mathcal{R}$.
- [S-τ]. We have:

$$\frac{([\texttt{!a}\{g, R\}]\, p, \nu) \xrightarrow{\texttt{!a}} (p, \nu[R]) \quad (q, \eta) \xrightarrow{\texttt{?a}} (q', \eta[S])}{(p, \nu) \mid (q, \eta) \xrightarrow{\tau} (p', \nu') \mid (q', \eta') = s_1'}$$

where the premise requires $q = \texttt{?a}\{f, S\}.q' + q''$, with $\nu \in [\![g]\!]$ and $\eta \in [\![f]\!]$. Since $\eta \in [\![f]\!]$, by rule [M-+] we have:

$$(p, [\texttt{!a}], \nu) \parallel (\texttt{?a}\{g, R\}.\, q' + q'', [], \eta) \xrightarrow{\mathsf{B}:\texttt{?a}} (p, [], \nu) \parallel (q', [], \eta[R]) = s_2'$$

Then, $(s_1', s_2') \in \mathcal{R}_2 \subseteq \mathcal{R}$.

- [S-Del]. This case does not apply, since one buffers is not empty.

(b) To prove item (b) of Definition 36, we consider the possible moves of s_2:
- [M-⊕]. This case does not apply, given the form of s_2.
- [M-+]. We have:

$$s_2 \xrightarrow{\mathsf{B}:?\mathtt{a}} (p, [], \nu[R]) \parallel (q', [], \eta[S]) = s_2'$$

which requires $q = ?\mathtt{a}\{f, S\}.q' + q''$, with $\eta \in [\![f]\!]$. Since by hypothesis $\nu \in [\![g]\!]$, by rule [S-⊕] we have:

$$\frac{\overline{([!\mathtt{a}\{g, R\}]\, p, \nu) \xrightarrow{!\mathtt{a}} (p, \nu[R])}\ [\oplus] \qquad \overline{(?\mathtt{a}\{f, S\}.q' + q'', \eta) \xrightarrow{?\mathtt{a}} (q', \eta[S])}\ [+]}{s_1 \xrightarrow{\tau} (p, \nu[R]) \mid (q', \eta[S]) = s_1'}$$

and the thesis follows because $(s_1', s_2') \in \mathcal{R}_1 \subseteq \mathcal{R}$.
- [M-Del]. This case does not apply, given the form of s_2.

(c) To prove item (c) of Definition 36, assume that $s_2 \in S_2$. By definition of S_2, s_2 has the form $(\mathbf{1}, [], \nu) \parallel (q, [], \eta)$. Then, this case does not apply.

Case 3: Let $s_1 = ([!\mathtt{a}\{g, R\}]\, p \mid [!\mathtt{b}\{f, S\}]\, q, \nu)$, and let $s_2 = (p, [!\mathtt{a}], \nu[R]) \parallel (!\mathtt{b}\{f, S\}.q \oplus q', [], \eta)$. The thesis follows trivially, since both s_1 and s_2 are stuck, and neither of them is a success state.

Part B: s_1 turn-simulates s_2 via $\mathcal{R}$.
Case 1: Let $s_1 = (p, \nu) \mid (q, \eta)$ and $s_2 = (p, [], \nu) \parallel (q, [], \eta)$.

(a) To prove item (a) of Definition 36, we consider the possible moves of s_2:
- [M-⊕]. We have:

$$s_2 \xrightarrow{\mathsf{A}:!\mathtt{a}} (p', [!\mathtt{a}], \nu[R]) \parallel (q, [], \eta) = s_2'$$

where the premise requires $p = !\mathtt{a}\{g, R\}.p' \oplus p''$ and $\nu \in [\![g]\!]$. Hence, by rule [S-⊕] we have:

$$\frac{(p, \nu) \xrightarrow{\tau} ([!\mathtt{a}\{g, R\}]\, p', \nu)}{s_1 \xrightarrow{\tau} ([!\mathtt{a}\{g, R\}]\, p', \nu) \mid (q, \eta) = s_1'}$$

Then, $(s_1', s_2') \in \mathcal{R}_2 \subseteq \mathcal{R}$.
- [M-+]. This case does not apply.
- [M-Del]. We have:

$$s_2 \xrightarrow{\delta} (p, [], \nu + \delta \parallel q, [], \eta + \delta)$$

where the premise requires $\nu + \delta \in \mathtt{rdy}(p)$ and $\nu + \delta \in \mathtt{rdy}(q)$. Hence, by [Del] we have:

$$\frac{(p, \nu) \xrightarrow{\delta} (p, \nu + \delta) \qquad (q, \eta) \xrightarrow{\delta} (q, \eta + \delta)}{s_1 \xrightarrow{\delta} (p, \nu + \delta) \mid (q, \eta + \delta) = s_1'}$$

Then, $(s_1', s_2') \in \mathcal{R}$.

(b) To prove item (b) of Definition 36, we consider the possible moves of s_1:

- [S-$\oplus$]. We have:
$$s_1 \xrightarrow{\oplus} ([\mathtt{!a}\{g, R\}]\, p', \nu[R]) \mid (q, \eta)$$
whose premise requires $p = \mathtt{!a}\{g, R\}.p' \oplus p''$, with $\nu \in [\![g]\!]$. Hence $s_2 \xrightarrow{\mathsf{A:!a}} \!\!\!\!\rightarrow$.
- [S-τ]. This case does not apply.
- [S-Del]. We have:
$$s_1 \xrightarrow{\delta} (p, \nu + \delta) \mid (q, \eta + \delta)$$
where the premise requires $\nu + \delta \in \mathtt{rdy}(p)$ and $\nu + \delta \in \mathtt{rdy}(q)$. Hence, by [M-Del] we have $s_2 \xrightarrow{\delta} \!\!\!\!\rightarrow$.

(c) To prove item (b) of Definition 36, assume that $s_2 \in S_2$. By definition of $S - 2$, s_2 has the form $(\mathbf{1} \mid \mathbf{1}, [], \nu)$. Then $s_1 = (\mathbf{1}, \nu) \mid (\mathbf{1}, \eta) \in S_1$.

Case 2: Let $s_1 = ([\mathtt{!a}\{g, R\}]\, p, \nu) \mid (q, \eta)$, $s_2 = (p, [\mathtt{!a}], \nu[R]) \parallel (q, [], \eta)$, and $\nu \in [\![g]\!]$.

(a) To prove item (a) of Definition 36, we consider the possible moves of s_2:

- [M-$\oplus$]. This case does not apply.
- [M-+]. We have:
$$s_2 \xrightarrow{\mathsf{B:?a}} \!\!\!\!\rightarrow (p, [], \nu[R]), \parallel (q', [], \eta[S]) = s_2'$$
whose premise requires $q = \mathtt{?a}\{f, S\}.q' + q''$ with $\eta \in [\![f]\!]$. Since by hypothesis $\nu \in [\![g]\!]$, by [S-τ] we have:
$$\frac{([\mathtt{!a}\{g, R\}]\, p, \nu) \xrightarrow{\mathtt{!a}} (p, \nu[R]) \quad (q, \eta) \xrightarrow{\mathtt{?a}} (q', \eta[S])}{(p, \nu) \mid (q, \eta) \xrightarrow{\tau} (p', \nu') \mid (q', \eta') = s_1'}$$
Then, $(s_1', s_2') \in \mathcal{R}$.
- [M-Del]. This case does not apply.

(b) To prove item (b) of Definition 36, we consider the possible moves of s_1:

- [S-$\oplus$]. We have:
$$s_1 \xrightarrow{\oplus} ([\mathtt{!a}\{g, R\}]\, p, \nu) \mid ([\mathtt{!b}\{f, S\}]\, q, \eta) = s_1'$$
whose premise requires $q = \mathtt{!b}\{f, S\}.q' \oplus q''$ and $\nu \in [\![f]\!]$. We have that s_2 is stuck but so is s_1'; and $(s_1', s_2) \in \mathcal{R}$.
- [S-τ]. We have:
$$s_1 \xrightarrow{\tau} (p, \nu[R]) \mid (q', \eta[S])$$
whose premise requires $q = \mathtt{?b}\{f, S\}.q' + q''$ and $\eta \in [\![f]\!]$. Hence $s_2 \xrightarrow{\mathsf{B:?a}} \!\!\!\!\rightarrow$ $(p \parallel q', [], \nu[R] \sqcup \eta[S]) = s_2'$ and $(s_1', s_2') \in \mathcal{R}$.
- [S-Del]. This case does not apply.

(c) To prove c, let us assume $s_2 \in S_2$, which implies $s_2 = (\mathbf{1} \mid \mathbf{1}, \nu)$. By hypothesis of *case 2* this is not possible.

Case 3: Let $s_1 = ([!\mathtt{a}\{g, R\}]\, p, \nu) \mid ([!\mathtt{b}\{f, S\}]\, q, \eta)$, $s_2 = (p, [!\mathtt{a}], \nu[R]) \parallel (!\mathtt{b}\{f, S\}.q \oplus q'), [], \eta)$. In this case, both s_1 and s_2 are stuck and neither of them is a success state. □

Theorem 9. $\bowtie \; = \bowtie_M$.

Proof. For the inclusion $\subseteq$, assume that $p \bowtie q$, i.e. $p(\lhd \cap \rhd)q$. By Lemma 21, $(p, \nu_0) \mid (q, \eta_0)$ is turn-bisimilar to $(p, [], \nu_0) \parallel (q, [], \eta_0)$. By $(p, [], \nu_0) \parallel (q, [], \eta_0) \preccurlyeq (p, \nu_0) \mid (q, \eta_0)$ and $(p, \nu_0)(\lhd \cap \rhd)(q, \nu_0)$, Lemma 20 implies that $(p, [], \nu_0)(\lhd_M \cap \rhd_M)(q, [], \eta_0)$, hence $p \bowtie_M q$. The other inclusion is similar. □

Lemma 2. *Let* $\gamma_0 = (p, [], \nu_0) \parallel (q, [], \eta_0)$. *If* $p \bowtie q$, *then* $(\twoheadrightarrow_M, \gamma_0)$ *is deterministic, and for all finite timed traces* $\boldsymbol{\lambda}$ *there exists (unique)* γ *such that* $\gamma_0 \xrightarrow{\boldsymbol{\lambda}}\!\!\!\!\twoheadrightarrow_M \gamma$.

Proof. Simple inspection of the rules in Figure 4. □