TIMED SESSION TYPES

MASSIMO BARTOLETTI, TIZIANA CIMOLI, AND MAURIZIO MURGIA

Università degli Studi di Cagliari, Italy
e-mail address: bart@unica.it

Università degli Studi di Cagliari, Italy
e-mail address: t.cimoli@unica.it

Università degli Studi di Cagliari, Italy and University of Kent, UK
e-mail address: M.Murgia@kent.ac.uk

Abstract. Timed session types formalise timed communication protocols between two participants at the endpoints of a session. They feature a decidable compliance relation, which generalises to the timed setting the progress-based compliance between untimed session types. We show a sound and complete technique to decide when a timed session type admits a compliant one. Then, we show how to construct the most precise session type compliant with a given one, according to the subtyping preorder induced by compliance. Decidability of subtyping follows from these results.

1. Introduction

Session types are formal descriptions of interaction protocols involving two or more participants over a network [31, 41]. 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 [7, 21, 26, 27, 34] and of tools [3, 23, 44] based on them in the last few years.

2012 ACM CCS: [**Theory of computation**]: Models of computation; Concurrency – Semantics and reasoning – Program reasoning – Program specifications – Program verification; Semantics and reasoning – Program semantics – Operational semantics.

Key words and phrases: session types, timed systems, compliance.

Full version of an Extended Abstract presented at FORTE'15.

LOGICAL METHODS
IN COMPUTER SCIENCE

DOI:10.2168/LMCS-???

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, namely 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 [18, 39], with the goal of extending compositional verification techniques [32, 33]. These approaches introduce time into an already sophisticated setting, featuring multiparty session types with asynchronous communication (via unbounded buffers).

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, would enable the implementation of programming tools and infrastructures for the development of safe communication-oriented distributed applications.

Timed session types. In this paper we present a theory of binary timed session types (TSTs). For instance, we describe as the following TST the protocol of a service taking as input a zip code, and then either providing as output the current weather, or aborting:

$$p \;=\; \texttt{?zip}\{x\}.\,(\texttt{!weather}\{5 < x < 10\} \oplus \texttt{!abort}\{x < 1\}) \tag{1.1}$$

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 send `weather` in a time window of $(5, 10)$ time units, or `abort` the protocol within 1 time unit.

A standard notion of compliance in the untimed setting is the one which relates two session types whenever their interaction never reaches a deadlock [11]. For instance, consider the two session types:

$$\texttt{?zip}.\,(\texttt{!weather} \oplus \texttt{!abort}) \qquad\qquad \texttt{!zip}.\,(\texttt{?weather} + \texttt{?abort}) \tag{1.2}$$

After the synchronisation on `zip`, the leftmost session type can internally choose between one of the outputs `weather` and `abort`, and each one of these choices is matched by the external choice in the rightmost session type. Therefore, the two session types in (1.2) are considered compliant. In the *untimed* setting, the notion of compliance is decidable: a procedure to decide whether two session types are compliant can just explore the finite-state LTS which describes their interaction, and return true whenever there are not deadlocks.

In the timed setting, we can use the same notion of compliance as in the untimed case, but applied on the *timed* LTS that describes the interaction of the two TSTs. For instance, the TST p in (1.1) is *not* compliant with:

$$q \;=\; \texttt{!zip}\{y\}.\,(\texttt{?weather}\{y < 7\} + \texttt{?abort}\{y < 5\}) \tag{1.3}$$

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). Differently from the untimed

case, the timed LTS of TSTs is infinite-state, hence compliance is not trivially decidable. A relevant problem is then the decidability of compliance in the timed case.

A further difference from the untimed case is that not every TST admits a compliant one. In the untimed case, a session type is always compliant to its syntactic dual, which is obtained by swapping outputs with inputs, and internal with external choices, as in (1.2). For instance, consider the client protocol:

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

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. This observation gives rise to other two questions. How to decide if a TST admits a compliant one? In such case, can we effectively construct a TST which is compliant with the given one?

Another relevant notion in the untimed setting is that of semantic subtyping [6]. Roughly, a session type p' is a subtype of p whenever all the session types which are compliant with p are compliant also with p'. This notion can be used to detect when a service can be safely substituted by another one. Indeed, 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. Establishing decidability of subtyping for TSTs would allow to check substitutability of time-aware services.

Contributions. We summarize the main contributions of the present work as follows:

(1) We give the syntax and semantics of TSTs. Their semantics is a conservative extension of the synchronous semantics of untimed session types [6], adding *clock valuations* to associate each clock with a positive real.
(2) We develop a sound and complete decision procedure for verifying compliance between TSTs (Theorem 3.7). To do that, we reduce this problem to that of model-checking deadlock freedom in timed automata [2], which is decidable.
(3) We develop a procedure to detect whether a TST admits a compliant one. This procedure takes the form of a kind system which associates, to each TST p, a set of clock valuations under which p admits a compliant. The kind system is sound and complete (Theorem 4.6 and Theorem 4.8), and kind inference is decidable (Theorem 5.7). From this we infer a decidable (sound and complete) procedure for the existence of a compliant.
(4) We exploit the kind system to define the *canonical compliant* of a TST (Definition 4.5). From the decidability results on the kind system we obtain an effective procedure to construct the canonical compliant (Theorem 5.8).
(5) We study the semantic subtyping preorder for TSTs. Building upon the decidability of compliance and of kind inference, we prove that semantic subtyping between TSTs is decidable (Theorem 6.4). We also show that the canonical compliant of a TST is the *greatest* TST compliant with the original one, according to the preorder (Theorem 6.2).
(6) We provide TSTs with a *monitoring semantics* (Definition 9.1), which detects when a participant does not respect its TST. This semantics is deterministic (Lemma 9.2), and it is coherent with the semantics of item (1) with respect to compliance (Theorem 9.9). Further, 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 (Lemma 9.4).
(7) We develop a suite of tools which implement the primitives discussed above, including a compliance checker, the canonical compliant construction, and an execution monitor based on TSTs. Our tools are available at co2.unica.it.

Applications of timed session types. A theory of timed session types with the primitives outlined above can be applied to the design and implementation of distributed applications. For instance, the message-oriented middleware in [9] exploits our theory to allow disciplined interactions between mutually distrusting services. 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.

The middleware guarantees that sessions are established only between services with compliant TSTs. More specifically, assume that a service A advertises a contract p to the store (this is only possible if p admits a compliant). A session between A and another service 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 canonical compliant of p). In the first case, the middleware exploits the decision procedure for compliance (Theorem 3.7), while in the second one it uses the canonical compliant construction (Definition 4.5).

The middleware also implements an execution monitor for TSTs, to check if the actions performed by services conform to their protocol, and — otherwise — to detect which services have caused the violation. More specifically, 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 9.1), and possible misbehaviours are detected (according to Definition 9.3) and sanctioned. Sanctions negatively affect the reputation of a service, and consequently its chances to establish new sessions.

In systems of *honest* services that always respect their contracts, compliance ensures progress of the whole application. In systems with dishonest services, dynamic verification of all the exchanged messages guarantees safe executions, and the sanction mechanism automatically marginalizes the dishonest services.

The middleware APIs (available at co2.unica.it) implement the following primitives: (a) `tell` advertises a TST (say, p) to the middleware. Firstly, the middleware checks if p admits a compliant one (otherwise, it rejects p). Then, it searches the store for a TST compliant with p. When such TST is found, the middleware starts a new session between the respective services, sending them the session identifier. (b) `accept` is used by a service A to directly establish a session with a service B, knowing the identifier of a TST p advertised by B. In this case A does not advertise a TST, and its contract in the new session will be the canonical compliant of p. (c) `send` and `receive` perform outputs and inputs of messages in an already established session. The middleware monitors all the exchanged messages, also keeping track of the passing of time, to detect and sanction contract violations.

Structure of the paper. We start in Section 2 by giving the syntax and semantics of TSTs. In Section 3 we define a compliance relation between TSTs, extending to the timed setting the progress-based notion of compliance between untimed session types. In Section 4 we study when a TST admits a compliant one, and we define a construction to obtain the canonical compliant of a TST. Decidability of this construction is established in Section 5. In Section 6 we study the semantic subtyping preorder for TSTs. In Section 7 we report a case study of a user agreement policy, showing how to model it as a TST. We detail the encoding of TSTs into timed automata in Section 8. In Section 9 we address the problem of dynamically monitoring interactions regulated by TSTs. Finally, in Section 10 we discuss some related work. The proofs of all our statements are either in the main text, or in Appendices A to E.

2. Timed session types: syntax and semantics

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 the set $\mathbb{N}$ of natural numbers. 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 2.1 (**Guard**). We define the set $\mathcal{G}_{\mathbb{C}}$ of *guards* over clocks $\mathbb{C}$ as follows:

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

A TST p models the behaviour of a single participant involved in an interaction (Definition 2.2). To give some intuition, we consider two participants, Alice (A) and Bob (B), who want to interact. A 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. Dually, B advertises an *external choice* $\sum_i ?\mathtt{a}_i\{g_i, R_i\}\,.\,q_i$ to declare 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 2.2 (**Timed session type**). *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 (e.g., we forbid both $\mathrm{rec}\, X\,.\,X$ and $\mathrm{rec}\, X\,.\,\mathrm{rec}\, Y\,.\,p$).

Except where 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* $\mathbf{1}$ can be omitted.

Example 2.3 (**Simplified PayPal**). 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 Section 7 for the full version). PayNow helps customers in on-line purchasing, providing protection against seller 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.

PayNow's agreement 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 2.7). This relation describes all and only the *correct* interactions: e.g., 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 9 we will study another semantics, which can also describe the behaviour of *dishonest* participants who do not respect their protocols.

Definition 2.4 (**Clock valuations**)**.** We denote with $\mathbb{V} = \mathbb{C} \to \mathbb{R}_{\geq 0}$ the set of *clock valuations*. We use meta-variables $\nu, \eta, \ldots$ to range over $\mathbb{V}$, and we denote with ν_0, η_0 the *initial* clock valuations, which map each clock to zero. Given a clock valuation ν, we define the following valuations:

- $\nu + \delta$ increases each clock in ν by the delay $\delta \in \mathbb{R}_{\geq 0}$, i.e.:
$$(\nu + \delta)(t) = \nu(t) + \delta \qquad \text{for all } t \in \mathbb{C}$$
- $\nu[R]$ *resets* all the clocks in the set $R \subseteq \mathbb{C}$, i.e.:
$$\nu[R](t) = \begin{cases} 0 & \text{if } t \in R \\ \nu(t) & \text{otherwise} \end{cases}$$

Definition 2.5 (**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 [16] some basic operations on *sets* of clock valuations (ranged over by $\mathcal{K}, \mathcal{K}', \ldots \subseteq \mathbb{V}$).

Definition 2.6 (**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:

$$\downarrow \mathcal{K} \;=\; \{\nu \mid \exists \delta \geq 0 : \nu + \delta \in \mathcal{K}\} \qquad\qquad \mathcal{K}[T]^{-1} \;=\; \{\nu \mid \nu[T] \in \mathcal{K}\}$$

Definition 2.7 (**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 a labelled relation $\to$ over configurations (Figure 1), whose labels are either silent actions τ, delays δ, or branch labels, and where we define the set of clock valuations $\texttt{rdy}(p)$ as:

$$\texttt{rdy}(p) \;=\; \begin{cases} \downarrow \bigcup [\![g_i]\!] & \text{if } p = \bigoplus_{i \in I} \texttt{!a}_i\{g_i, R_i\}\,.\,p_i \\ \mathbb{V} & \text{if } p = \sum \cdots \text{ or } p = \mathbf{1} \\ \emptyset & \text{otherwise} \end{cases}$$

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

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

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

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

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

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

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

As usual, we write $p \xrightarrow{\alpha} p'$ as a shorthand for $(p,\alpha,p') \in \rightarrow$, with $\alpha \in \mathtt{L} \cup \{\tau\} \cup \mathbb{R}_{\geq 0}$. Given a relation $\rightarrow$, we denote with $\rightarrow^*$ its reflexive and transitive closure.

We now comment the rules in Figure 1. The first four rules 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 guard is satisfied in the clock valuation ν. This results in the term $[\texttt{!a}\{g,R\}]\,p$, which can only fire $\texttt{!a}$ through rule [!], without making time pass. This term represents a state where the endpoint has committed to branch $\texttt{!a}$ in a specific time instant[1]. 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 $\texttt{rdy}$. The last three rules deal with configurations. Rule [S-$\oplus$] allows a TST to commit in an internal choice. Rule [S-τ] is the standard synchronisation rule *à la* CCS. Rule [S-Del] allows time to pass, equally for both endpoints.

Example 2.8. Let $p = \texttt{!a} \oplus \texttt{!b}\{t > 2\}$, let $q = \texttt{?b}\{t > 5\}$, and consider the computations:

$$(p,\nu_0) \mid (q,\eta_0) \xrightarrow{7} \xrightarrow{\tau} ([\texttt{!b}\{t>2\}]\,,\nu_0+7) \mid (q,\eta_0+7) \xrightarrow{\tau} (\mathbf{1},\nu_0+7) \mid (\mathbf{1},\eta_0+7) \qquad (2.1)$$

$$(p,\nu_0) \mid (q,\eta_0) \xrightarrow{\delta} \xrightarrow{\tau} ([\texttt{!a}]\,,\nu_0+\delta) \mid (q,\eta_0+\delta) \qquad (2.2)$$

$$(p,\nu_0) \mid (q,\eta_0) \xrightarrow{3} \xrightarrow{\tau} ([\texttt{!b}\{t>2\}]\,,\nu_0+3) \mid (q,\eta_0+3) \qquad (2.3)$$

The computation in (2.1) reaches success, while the other two computations reach a deadlock state. In (2.2), p commits to the choice $\texttt{!a}$ 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 $\texttt{?a}$). In (2.3), p commits to $\texttt{!b}$ 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

[1] This is quite similar to the handling of internal choices in [5, 10, 13]. In these works, an internal choice $\texttt{!a}.p \oplus q$ first commits to one of the branches (say, $\texttt{!a}.p$) through an internal action, taking a transition to a singleton internal choice $\texttt{!a}.p$. In this state, only the action $\texttt{!a}$ is enabled — as in our $[\texttt{!a}\{g,R\}]\,p$.

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.

Note that, even when p and q have shared clocks, the rules in Figure 1 ensure that there is no interference between them. For instance, if a transition of (p, ν) resets some clock t, this has no effect on a clock with the same name in q, i.e. on a transition of $(p, \nu) \mid (q, \eta)$. Thus, without loss of generality we will assume that the clocks in p and in q are disjoint.

3. Compliance between TSTs

We extend to the timed setting the standard progress-based compliance between (untimed) session types [6, 11, 22, 36]. 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, the interaction between services correctly implementing[2] them will progress (without communication and time errors), until both services reach a success state.

Definition 3.1 (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:

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

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

Note that item (ii) of the definition of deadlock can be equivalently phrased as follows: $(p, \nu) \mid (q, \eta) \not\xrightarrow{\tau}$ (i.e., the configuration cannot do a τ-move in the *current* clock valuation), and there does not exist any $\delta > 0$ such that $(p, \nu) \mid (q, \eta) \xrightarrow{\delta} \xrightarrow{\tau}$.

Example 3.2. The TSTs $p = \texttt{?a}\{t < 5\}.\texttt{!b}\{t < 3\}$ and $q = \texttt{!a}\{t < 2\}.\texttt{?b}\{t < 3\}$ are compliant, but p is *not* compliant with $q' = \texttt{!a}\{t < 5\}.\texttt{?b}\{t < 3\}$. Indeed, if q' outputs $\texttt{a}$ at, say, time 4, the configuration will reach a state where no actions are possible, and time cannot pass. This is a deadlocked state, according to Definition 3.1.

Example 3.3. Consider a customer of PayNow (Example 2.3) 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 p in Example 2.3:

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

Compliance between TSTs is 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 setting. For instance, the following example shows that a recursive internal choice can be compliant with a *non*-recursive external choice — which can never happen in untimed session types.

[2]The notion of "correct implementation" of a TST is orthogonal to the present work. A possible instance of this notion can be obtained by extending to the timed setting the *honesty* property of [12].

Example 3.4. Let $p = \mathrm{rec}\, X.\,(\texttt{!a} \oplus \texttt{!b}\{x \leq 1\}.\texttt{?c}.\,X)$, $q = \texttt{?a} + \texttt{?b}\{y \leq 1\}.\texttt{!c}\{y > 1\}.\texttt{?a}$. 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 3.5 and Lemma 3.6 below coinductively characterise compliance between TSTs, by extending to the timed setting the coinductive compliance for untimed session types in [5]. 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 3.5. 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) $p = \bigoplus_{i \in I} \texttt{!a}_i\{g_i, R_i\}.\,p_i \implies \nu \in \mathtt{rdy}(p) \ \wedge\ q = \sum_{j \in J} \texttt{?a}_j\{g_j, R_j\}.\,q_j \ \wedge$
$\forall \delta, i:\ \nu + \delta \in [\![g_i]\!] \implies \exists j : \mathtt{a}_i = \mathtt{a}_j \wedge \eta + \delta \in [\![g_j]\!] \wedge (p_i, \nu + \delta[R_i])\ \mathcal{R}\ (q_j, \eta + \delta[R_j])$

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

Lemma 3.6. $p \bowtie q \iff \exists\, \mathcal{R}$ *coinductive compliance* $:\ (p, \nu_0)\ \mathcal{R}\ (q, \eta_0)$

Proof. See page 30. □

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.

Theorem 3.7. *Compliance between TSTs is decidable.*

Proof. We defer the actual proof after Theorem 8.15 in Section 8. □

4. Admissibility of a compliant

In the untimed setting, each session type p admits a compliant, i.e. there exists some q such that $p \bowtie q$. For instance, we can compute q by simply swapping internal choices with external ones (and inputs with outputs) in p (this q is called the *canonical dual* of p in some papers [21, 29]). 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 4.1. Consider the following TSTs:

$$p_1 = \texttt{!a}\{x \leq 2\}.\,\texttt{!b}\{x \leq 1\} \qquad p_2 = \texttt{!a}\{x \leq 2\} \oplus \texttt{!b}\{x \leq 1\}.\,\texttt{?a}\{x \leq 0\}$$

$$p_3 = \mathrm{rec}\, X.\,\texttt{?a}\{x \leq 1 \wedge y \leq 1\}.\,\texttt{!a}\{x \leq 1, \{x\}\}.\,X$$

The TST p_1 is not compliant with its naïve dual $q_1 = \texttt{?a}\{x \leq 2\}.\,\texttt{?b}\{x \leq 1\}$: even though q_1 can do the input $\texttt{?a}$ in the required time window, p_1 cannot perform $\texttt{!b}$ if $\texttt{!a}$ is performed after 1 time unit. For this very reason, no TST is compliant with p_1. Note instead that $q_1 \bowtie \texttt{!a}\{x \leq 1\}.\,\texttt{!b}\{x \leq 1\}$, which is *not* its naïve dual. In p_2, a similar deadlock situation occurs if the $\texttt{!b}$ 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

$$\Gamma \vdash \mathbf{1} : \mathbb{V} \qquad \text{[T-1]}$$

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

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

$$\Gamma, X : \mathcal{K} \vdash X : \mathcal{K} \qquad \text{[T-Var]}$$

$$\frac{\Gamma, X : \mathcal{K} \vdash p : \mathcal{K}'}{\Gamma \vdash \mathrm{rec}\, X \,.\, p : \bigcup \{ \mathcal{K}_0 \mid \exists \mathcal{K}_1 : \ \Gamma, X : \mathcal{K}_0 \vdash p : \mathcal{K}_1 \wedge \mathcal{K}_0 \subseteq \mathcal{K}_1 \}} \qquad \text{[T-Rec]}$$

Figure 2: Kind system for TSTs.

reaches the value 1; after this point, the guard $y \leq 1$ can no longer be satisfied, and then p_3 reaches a deadlock.

To establish when a TST admits a compliant, we define a kind system which associates to each p a set of clock valuations $\mathcal{K}$ (called *kind of* p). The kind of a TST is unique, and each closed TST is kindable (Theorem 4.4). If p has kind $\mathcal{K}$, then there exists some q such that, for all $\nu \in \mathcal{K}$, the configuration $(p, \nu) \mid (q, \nu)$ never reaches a deadlock (Theorem 4.6). Also the converse statement holds: if, for some q, $(p, \nu) \mid (q, \nu)$ never reaches a deadlock, then $\nu \in \mathcal{K}$ (Theorem 4.8). Therefore, p admits a compliant whenever the initial clock valuation ν_0 belongs to $\mathcal{K}$. We give a constructive proof of the correctness of the kind system, by showing a TST $\mathrm{co}(p)$ which we call the *canonical compliant* of p.

Definition 4.2 (Kind system for TSTs). 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 a compliant in every ν: indeed, $\mathbf{1}$ is compliant with itself. The kind of an external 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-$\oplus$], 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).

Example 4.3. Recall p_2 from Example 4.1. We have the following kinding derivation:

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

where $\mathcal{K} = [\![(x > 1) \wedge (x \leq 2)]\!]$. As noted in Example 4.1, intuitively p_2 has no compliant; this will be asserted by Theorem 4.8 below, as a consequence of the fact that $\nu_0 \notin \mathcal{K}$. However, since $\mathcal{K}$ is non-empty, Theorem 4.6 guarantees that there exists q such that $(p_2, \nu) \bowtie (q, \nu)$, for all clock valuations $\nu \in \mathcal{K}$.

The following theorem states that *every* closed TST is kindable, as well as uniqueness of kinding. We stress that being kindable does not imply admitting a compliant: this holds

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

Figure 3: Canonical compliant of a TST.

if and only if the initial clock valuation ν_0 belongs to the kind. Note that uniqueness of kinding holds at the *semantic* level, but the same kind can be represented syntactically in different ways. In Section 5 we show that uniqueness of kinding may be obtained also at the *syntactic* level, by representing kinds as guards in normal form [16].

Theorem 4.4 (**Uniqueness of kinding**). *For all* p *and* Γ *with* $\mathrm{fv}(p) \subseteq \mathrm{dom}(\Gamma)$*, there exists unique* $\mathcal{K}$ *such that* $\Gamma \vdash p : \mathcal{K}$*.*

Proof. See page 30. □

By exploiting the kind system we 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.

Definition 4.5 (**Canonical compliant**). For all kinding environments Γ and p kindable in Γ, we define the TST $\mathrm{co}_\Gamma(p)$ in Figure 3. We will abbreviate $\mathrm{co}_\Gamma(p)$ as $\mathrm{co}(p)$ when $\Gamma = \emptyset$.

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

Theorem 4.6 (**Soundness**). *If* $\vdash p : \mathcal{K}$ *and* $\nu \in \mathcal{K}$*, then* $(p, \nu) \bowtie (\mathrm{co}(p)\,, \nu)$*.*

Proof. See page 33. □

Example 4.7. Recall $q_1 = \mathtt{?a}\{x \le 2\}.\,\mathtt{?b}\{x \le 1\}$ from Example 4.1. We have $\mathrm{co}(q_1) = \mathtt{!a}\{x \le 1\}.\,\mathtt{!b}\{x \le 1\}$. Since $\vdash q_1 : \mathcal{K} = [\![x \le 1]\!]$ and $\nu_0 \in \mathcal{K}$, by Theorem 4.6 we have that $q_1 \bowtie \mathrm{co}(q_1)$, as anticipated in Example 4.1.

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

Proof. See page 35. □

Compliance is not transitive, in general: however, Theorem 4.10 below states that transitivity holds when passing through the canonical compliant.

Lemma 4.9. *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. See page 35. □

Theorem 4.10 (**Transitivity of compliance**). *If* $p \bowtie p'$ *and* $\mathrm{co}(p') \bowtie q$*, then* $p \bowtie q$*.*

Proof. Straightforward after Lemma 4.9. □

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

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

$$\Gamma \vdash_I \mathrm{rec}\, X.\, p : \textstyle\prod_{i\geq 0} \hat{F}^i_{\Gamma,X,p}(\mathbb{V}) \quad \text{[I-Rec]} \qquad \textit{where } \hat{F}_{\Gamma,X,p}(\mathcal{K}) = \mathcal{K}' \text{ iff } \Gamma, X : \mathcal{K} \vdash_I p : \mathcal{K}'$$

Figure 4: Kind inference rules.

5. Computability of the canonical compliant

In this section we show that the canonical compliant construction is computable. To prove this, we first show the decidability of kind inference (Theorem 5.7). This fact is not completely obvious, because the cardinality of the set of kinds is $2^{2^{\aleph_0}}$; however, the kinds constructed by our inference rules can always be represented syntactically by guards.

To prove the decidability of kind inference we will use the fundamental notion of clock region, introduced in [2] (although our definition is more similar to the one in [30]).

Definition 5.1 (Region). For all $d \in \mathbb{N}$, a *d-region* is a set of clock valuations $\mathcal{K}$ where:
(1) there exists a guard g, with all constants bounded by d, such that $[\![g]\!] = \mathcal{K}$;
(2) if $\mathcal{K}$ properly contains a d-region $\mathcal{K}'$, then $\mathcal{K}'$ is empty.
We call *d-zones*[3] (ranged over by $\mathcal{Z}, \mathcal{Z}', \ldots$) the finite unions of d-regions.

It is well known that, for all d, the set of d-regions is finite [2], and so also the induced set of d-zones. Further, d-zones are closed under past, inverse reset, union, intersection and difference, and for every guard g with all constants below d, $[\![g]\!]$ is a d-zone [30, 16]. By the above, it follows immediately that the set of d-zones, ordered by set inclusion, forms a finite complete lattice. In our case, d will be the greatest constant in the TST under analysis.

The following lemma recalls some well known facts about functions over complete lattices and their fixed points. These facts will be used to prove that kind inference is decidable.

Lemma 5.2. *Let $(D, \leq)$ be a complete lattice with top $\top$, and let f be a monotonic endofunction over D. Then:*
(1) *if D is finite, then f is cocontinuous.*
(2) $\mathrm{gfp}(f) \leq \prod_i f^i(\top)$.
(3) *if f is cocontinuous, then* $\mathrm{gfp}(f) = \prod_i f^i(\top)$

Proof. See page 36. □

We start by showing that the kind obtained by rule [T-Rec] in Figure 2 can be characterised as the greatest fixed point (over the lattice $(2^{\mathbb{V}}, \subseteq)$) of a functional, defined below.

Definition 5.3. Given Γ, X, p, with X *not* in the domain of Γ, we define:

$$F_{\Gamma,X,p}(\mathcal{K}) = \mathcal{K}' \quad \text{whenever} \quad \Gamma, X : \mathcal{K} \vdash p : \mathcal{K}'$$

The subscript of $F_{\Gamma,X,p}$ will be omitted when clear from the context.

[3]The term "zone" is often referred to convex sets of clocks, while our definition includes non-convex ones.

Note that, by uniqueness of kinding, F is a function; further, Theorem 4.4 ensures that F is total when $\mathrm{fv}(p) \subseteq \mathrm{dom}(\Gamma) \cup \{X\}$. The following lemma states that F is monotonic. Then, the Knaster-Tarski fixed point theorem [42] ensures that [T-Rec] yields the gfp of F.

Lemma 5.4. *The function $F_{\Gamma,X,p}$ is monotonic, for all Γ, X, p with $\mathrm{fv}(p) \subseteq \mathrm{dom}(\Gamma) \cup \{X\}$.*

Proof. See page 36. □

To prove decidability of the kinding relation, we introduce in Figure 4 an alternative set of rules, with judgements of the form $\Gamma \vdash_I p : \mathcal{K}$. Lemma 5.6 below shows that the kind relations $\vdash$ and $\vdash_I$ are equivalent. The new set of rules can be exploited as a kind inference algorithm: in particular, rule [I-Rec] allows for computing the kind of a recursive TST $\mathrm{rec}\, X\,.\, p$ by evaluating the non-increasing sequence $\hat{F}^i_{\Gamma,X,p}(\mathbb{V})$ until it stabilizes. The following lemma states that the kinds inferred through the relation $\vdash_I$ are zones. By [16] it follows that the kind of a TST can always be represented as a guard.

Lemma 5.5. *If $\Gamma \vdash_I p : \mathcal{K}$, for some Γ which maps variables to zones, then $\mathcal{K}$ is a zone.*

Proof. Easy, by induction on the structure of p and inspection of the kind inference rules (exploiting finiteness of the set of zones). □

Lemma 5.6. *For all Γ mapping variables to zones, and for all p such that $\mathrm{fv}(p) \subseteq \mathrm{dom}(\Gamma)$:*

$$\Gamma \vdash p : \mathcal{K} \iff \Gamma \vdash_I p : \mathcal{K}$$

Proof. See page 37. □

Theorem 5.7. *Kind inference is decidable.*

Proof. By Lemma 5.6, kinds of closed TSTs can be inferred by the rules in Figure 4. All the operations between zones used in Figure 4 (except $\hat{F}$) are well know to be computable [30, 16]. By finiteness of the set of zones, also $\hat{F}$ is computable. □

The following theorem states that the canonical compliant construction is computable.

Theorem 5.8. *The function $\mathrm{co}(\cdot)$ is computable.*

Proof. It follows by the fact that all the operations in Definition 4.5 are computable. □

6. Subtyping

In this section we study the semantic subtyping preorder, which is a sound and complete model of the Gay and Hole subtyping relation (in reverse order) for untimed session types [6]. Intuitively, p is subtype of q if every q' compliant with q is compliant with p, too.

Definition 6.1 (**Semantic subtyping**)**.** For all TSTs p, we define the set $(p,\nu)^{\bowtie}$ as:

$$(p,\nu)^{\bowtie} \;=\; \{(q,\eta) \mid (p,\nu) \bowtie (q,\eta)\}$$

Then, we define the preorder $\sqsubseteq$ between TSTs as follows:

$$p \sqsubseteq q \qquad \text{whenever} \qquad (p,\nu_0)^{\bowtie} \supseteq (q,\nu_0)^{\bowtie}$$

The following theorem states that, as in the untimed setting, the canonical compliant of p is the maximum (i.e., the most "precise") in the set of TSTs compliant with p.

Theorem 6.2. $q \bowtie p \implies q \sqsubseteq \mathrm{co}(p)$

Proof. Assume that $q \bowtie p$ and $\vdash p : \mathcal{K}$, and let $(r, \eta) \in (\mathrm{co}(p), \nu_0)^{\bowtie}$. Since $(q, \nu_0) \bowtie (p, \nu_0)$, by Theorem 4.8 we obtain $\nu_0 \in \mathcal{K}$. Therefore, by $(\mathrm{co}(p), \nu_0) \bowtie (r, \eta)$ and Lemma 4.9 we conclude that $(q, \nu_0) \bowtie (r, \eta)$. So, $(r, \eta) \in (q, \nu_0)^{\bowtie}$, from which the thesis follows. □

The following theorem reduces the problem of deciding $p \sqsubseteq q$ to that of checking compliance between p and $\mathrm{co}(q)$, when q admits a compliant (otherwise $(q, \eta_0)^{\bowtie} = \emptyset$, so q is supertype of every p). Since compliance, the canonical compliant construction, and checking the admissibility of a compliant are all decidable (Theorem 3.7, Theorem 5.8), this implies decidability of subtyping (Theorem 6.4).

Theorem 6.3. *For all TSTs p, q:*

$$p \sqsubseteq q \iff \begin{cases} p \bowtie \mathrm{co}(q) & \text{if } q \text{ admits a compliant} \\ \mathit{true} & \text{otherwise} \end{cases}$$

Proof. If q does not admit a compliant then the thesis is trivial, so assume that $(q, \eta_0)^{\bowtie} \neq \emptyset$. For the ($\Rightarrow$) direction, assume that $p \sqsubseteq q$. Since q admits a compliant, by Theorem 4.8 there exists some $\mathcal{K}$ such that $\vdash q : \mathcal{K} \ni \nu_0$. By Theorem 4.6, it follows that $\mathrm{co}(q) \bowtie q$. Then, by Definition 6.1 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 4.10 we conclude that $q' \bowtie p$, from which the thesis follows. □

Theorem 6.4 (**Decidability of subtyping**)**.** *Subtyping between TSTs is decidable.*

Proof. Immediate consequence of Theorem 3.7 and Theorem 6.3. □

Unlike in the untimed case, the canonical compliant construction is not involutive, i.e. $\mathrm{co}(\mathrm{co}(p))$ is not equal to p, in general. However, p and $\mathrm{co}(\mathrm{co}(p))$ are still strongly related, as they have the same set of compliant TSTs, in every ν in the kind of p (Theorem 6.5). By Definition 6.1, this implies that $p \sqsubseteq\sqsupseteq \mathrm{co}(\mathrm{co}(p))$, for all kindable p.

Theorem 6.5. *Let $\vdash p : \mathcal{K}$ and $\nu \in \mathcal{K}$. Then: $(p, \nu)^{\bowtie} = (\mathrm{co}(\mathrm{co}(p)), \nu)^{\bowtie}$.*

Proof. Suppose that $\vdash p : \mathcal{K}$ and $\nu \in \mathcal{K}$. By Theorem 4.6:

$$(p, \nu) \bowtie (\mathrm{co}(p), \nu) \tag{6.1}$$

Assume that $\vdash \mathrm{co}(p) : \mathcal{K}'$. By (6.1) and Theorem 4.8 it follows that $\nu \in \mathcal{K}'$. By repeating the same argument twice, we also obtain that:

$$(\mathrm{co}(p), \nu) \bowtie (\mathrm{co}(\mathrm{co}(p)), \nu) \tag{6.2}$$

$$(\mathrm{co}(\mathrm{co}(p)), \nu) \bowtie (\mathrm{co}(\mathrm{co}(\mathrm{co}(p))), \nu) \tag{6.3}$$

To prove $(p, \nu)^{\bowtie} \subseteq (\mathrm{co}(\mathrm{co}(p)), \nu)^{\bowtie}$, let $(q, \eta) \in (p, \nu)^{\bowtie}$. By applying Lemma 4.9 on $(q, \eta) \bowtie (p, \nu)$ and on (6.2), we obtain $(q, \eta) \bowtie (\mathrm{co}(\mathrm{co}(p)), \nu)$.
To prove $(p, \nu)^{\bowtie} \supseteq (\mathrm{co}(\mathrm{co}(p)), \nu)^{\bowtie}$, let $(q, \eta) \in (\mathrm{co}(\mathrm{co}(p)), \nu)^{\bowtie}$. By applying Lemma 4.9 on (6.1) and (6.3) (and using commutativity of compliance), we obtain:

$$(\mathrm{co}(\mathrm{co}(\mathrm{co}(p))), \nu) \bowtie (p, \nu) \tag{6.4}$$

Finally, by applying Lemma 4.9 on $(q, \eta) \bowtie (\mathrm{co}(\mathrm{co}(p)), \nu)$ and on (6.4), we conclude that $(q, \eta) \bowtie (\mathrm{co}(\mathrm{co}(p)), \nu)$. □

7. Case study: Paypal User Agreement

As a case study, we formalise as a TST (part of) the "protection for buyers" section of the PayPal User Agreement [1], which regulates the interaction between Paypal and buyers in trouble during online purchases. When a buyer has not received the item they have paid for (`inr`), or if they have received something significantly different from what was described (`snad`), they can open a *dispute*. The dispute can be opened within 180 days ($t_{pay} < 180$) of the payment date (`pay`). After opening the dispute, the buyer and the seller may try to solve the problem, or it might be the case that the item finally arrives; otherwise, if an agreement (`ok`) is not found within 20 days ($t_{inr} < 20$), the buyer can escalate the dispute to a claim (`claimINR`,`claimSNAD`). However, in case of an item not received, the buyer must wait at least 7 days from the date of payment to escalate the dispute ($t_{pay} > 7$). Upon not reaching an agreement, if still the buyer does not escalate the dispute to a claim within 20 days ($t_{pay} > 20$), PayPal will close the dispute (`close`).

During the claim process, PayPal may require the buyer to provide documentation to support the claim, for instance receipts (`rcpt`) or photos (`photo`), and the buyer must comply in a *timely manner* to what they are required to do. For SNAD claims, if the claim is accepted, PayPal may require the buyer to ship the item back to the Seller, to PayPal, or to a third party and to provide proof of delivery. In case the item is counterfeit, the item will be destroyed (`destroy`) and not shipped back to the seller(`sendBack`). After that, the buyer will be refunded. In some cases, the buyer is not eligible for a refund (`notEligible`).

We can formalise this agreement as the following TST:

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

Let us consider a possible buyer Alice, who wants to see if she may entrust PayPal for her transactions. Alice is willing to wait 10 days to receive the item she has paid for, but after that she will open a claim. She will readily provide PayPal with every documentation they may need in order to issue the refund. In case she receives an item significantly different from what she has paid for, she will complain to PayPal by opening a claim as soon as the item is received. Alice will timely comply to do whatever PayPal requires (either to destroy the item or to send it back) in order to be refunded. Alice's requirements can be formalised as the following TST:

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

Alice's and PayPal's TSTs are compliant, according to Definition 3.1. However, we can see that PayPal's TST lacks some important details: what does it means *timely comply to what is required*? And, most importantly: how long will it take for a buyer to be refunded? Without a deadline on the `!refund` action, Alice may possibly wait forever.

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

$$\frac{(l_1,\texttt{!a},g_1,R_1,l_1') \in E_1 \quad (l_2,\texttt{?a},g_2,R_2,l_2') \in E_2 \quad \nu \in [\![g_1 \wedge g_2]\!] \quad \nu[R_1][R_2] \in [\![I_1(l_1') \wedge I_2(l_2')]\!]}{(l_1,l_2,\nu) \xrightarrow{\texttt{a}}_N (l_1',l_2',\nu[R_1][R_2])} \text{[TA2]}$$

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

Figure 5: Semantics of networks of TA (symmetric rules omitted).

8. Encoding TSTs into Timed Automata

We define a semantic-preserving encoding of TSTs into timed automata (Definition 8.12), and we exploit it to devise an effective procedure to decide compliance (Theorem 8.15).

8.1. **Timed automata.** A timed automaton (TA) [2] is a non-deterministic automaton with a finite set of clocks, used to put constraints on when to take an edge, or to stay in a location. We denote with $\mathtt{L}_\tau = \mathtt{A}^! \cup \mathtt{A}^? \cup \{\tau\}$ the set of labels, ranged over by $\ell_\tau, \ell_\tau', \ldots$.

Definition 8.1 (**Timed automaton** [2])**.** A TA is a tuple $A = (Loc, Loc^u, l_0, E, I)$ where: Loc is a finite set of *locations*; $Loc^u \subset Loc$ is the set of *urgent* locations; $l_0 \in Loc$ is the *initial* location; $E \subseteq Loc \times \mathtt{L}_\tau \times \mathcal{G}_\mathbb{C} \times \wp(\mathbb{C}) \times Loc$ is a set of edges; and $I : Loc \to \mathcal{G}_\mathbb{C}$ is the *invariant* function.

To define the semantics of TA, we consider a *network* of TA which can take internal transitions or synchronize on channels. For our purposes, we just need networks of *two* TA interacting through *binary* channels, which correspond to actions $\mathtt{a}, \mathtt{b}, \ldots$.

Definition 8.2 (**Semantics of TA**)**.** Let $A_i = (Loc_i, Loc_i^u, l_i^0, E_i, I_i)$ be TA, for $i \in \{1,2\}$. We define the behaviour of the network $A_1 \mid A_2$ as the timed LTS $(S, s_0, Lab, \to_N)$, where: $S = Loc_1 \times Loc_2 \times \mathbb{V}$; $s_0 = (l_1^0, l_2^0, \nu_0)$; $Lab = \mathtt{A} \cup \{\tau\} \cup \mathbb{R}_{\geq 0}$; $\to_N$ is defined in Figure 5.

The state of a network is given by the locations of its automata, and a clock valuation. Rule [TA1] allows one of the TA to take an internal edge, if its guard is satisfied and if, after resetting the clocks in R, the invariants of the target locations are satisfied. Rule [TA2] allows two TA to synchronize on a channel $\mathtt{a}$ if (i) their current locations are outbound edges with complementary labels (e.g., $\mathtt{!a}$ and $\mathtt{?a}$); (ii) the guards of those edges are satisfied by the clock valuation; and (iii) the invariants of the target locations are satisfied after the clocks reset. Rule [TA3] allows time to pass, if the locations in the current state are not urgent, and their invariants are satisfied after the time delay. Note that all the clocks progress with the same pace, and taking an edge is an instantaneous action.

If the current locations of a state have no outgoing edges, then such state is called *success*, while a state is called *deadlock* if it is not success and no *action*-transitions are possible (neither in the current clock valuation, nor in the future).

Definition 8.3 (**Deadlock freedom**)**.** We say that a network state s is *deadlock* whenever: (i) s is not *success*, and (ii) $\nexists \delta \geq 0,\ \mathtt{a}_\tau \in \mathtt{A} \cup \{\tau\} : s \xrightarrow{\delta}_N \xrightarrow{\mathtt{a}_\tau}_N$. A network is *deadlock-free* if none of its reachable states is deadlock.

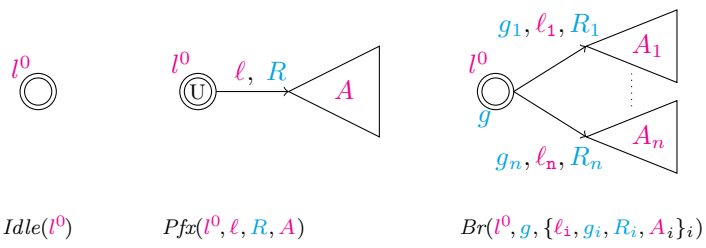

Figure 6: Patterns for TA composition, represented as in [28]. Circles denote locations (those marked with U are urgent), and arrows denote edges. Internal actions, `true` guards/invariants and empty resets are left blank. A TA is depicted as a triangle, whose left vertex represents its initial location (a double circle). An arrow from l^0 to triangle A represents an edge from l^0 to the initial location of A.

We now introduce some operators to compose TA. The *union* of a set of TA collects all the locations and all the edges; the invariant on a location is the conjunction of all the invariants defined on that location; the initial location is specified as a parameter. The *Idle* pattern creates a TA with only a success location; the *Pfx* pattern prefixes a location to a TA; finally, the *Br* pattern prefixes a location to a set of TA, using guarded edges.

Definition 8.4 (Union). For all $i \in I$, let $A_i = (Loc_i, Loc_i^u, l_i^0, E_i, I_i)$, and let $l \in \bigcup_i Loc_i$. We define $\bigsqcup_{i\in I}^{l} A_i = (\bigcup_i Loc_i, \bigcup_i Loc_i^u, l, \bigcup_i E_i, I)$, where $I(l_j) = \bigwedge_i I_i(l_j)$ for all $l_j \in \bigcup_i Loc_i$.

Definition 8.5 (Idle pattern). Let l^0 be a location. Then, $Idle(l^0) = (\{l^0\}, \emptyset, l^0, \emptyset, \emptyset)$.

Definition 8.6 (Prefix pattern). Let $A = (Loc_1, Loc_1^u, l_1^0, E_1, I_1)$ be a TA, let $\ell_\tau \in \mathtt{L}_\tau$, $l^0 \notin Loc_1$, and $R \subseteq \mathbb{C}$. Then, $Pfx(l^0, \ell_\tau, R, A) = (Loc_1 \cup \{l^0\}, Loc_1^u \cup \{l^0\}, l^0, E, I_1\{\mathtt{true}/l^0\})$, where $E = E_1 \cup \{(l^0, \ell_\tau, \mathtt{true}, R, l_1^0)\}$.

Definition 8.7 (Branch pattern). For all $i \in I$, let $A_i = (Loc_i, Loc_i^u, l_i^0, E_i, I_i)$, and let $g_i, g \in \mathcal{G}_\mathbb{C}$, $R_i \subseteq \mathbb{C}$, $\ell_{\tau_i} \in \mathtt{L}_\tau$. Then, $Br(l^0, g, \{(\ell_{\tau_i}, g_i, R_i, A_i) \mid i \in I\}) = (Loc, Loc^u, l^0, E, I)$, where: $Loc = \bigcup_i Loc_i \cup \{l^0\}$, $Loc^u = \bigcup_i Loc_i^u$, $E = \bigcup_i E_i \cup \bigcup_i \{(l^0, \ell_{\tau_i}, g_i, R_i, l_i^0)\}$, and $I = \bigcup_i I_i \cup \{(l^0, g)\}$.

8.2. **Defining equations.** The first step of our encoding from TSTs to TA is to put TSTs in a normal form where recursive terms $\mathrm{rec}\, X.\, p$ are replaced by *defining equations*. This alternative representation of infinite-state processes is quite common in concurrency theory [38], hence we will defer some standard technicalities to Appendix D. In our normal form (called DE-TST) each process is represented as a pair, composed of 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 where every recursion variable is guarded by exactly one action (Definition 8.8).

Definition 8.8 (Defining-equation normal form). A DE-TST is a pair (X, D), where D is a set of defining equations of the form $X' \triangleq p$, where p has the following syntax:

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

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

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

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

$$\frac{(X \triangleq \textstyle\sum \ldots) \in D \ \vee\ (X \triangleq \mathbf{1}) \in D}{(X,\ \nu) \xrightarrow{\delta}_D (X, \nu + \delta)}\ [\text{DE-Del1}] \qquad \frac{X \triangleq p \in D \quad \nu + \delta \in \mathtt{rdy}(p)}{(\tau X,\ \nu) \xrightarrow{\delta}_D (\tau X,\ \nu + \delta)}\ [\text{DE-Del2}]$$

$$\frac{(s, \nu) \xrightarrow{\tau}_D (s', \nu)}{(s, \nu) \mid (t, \eta) \xrightarrow{\tau}_D (s', \nu) \mid (t, \eta)}\ [\text{DES-}\oplus] \qquad \frac{(s, \nu) \xrightarrow{\delta}_D (s, \nu') \quad (t, \eta) \xrightarrow{\delta}_D (t, \eta')}{(s, \nu) \mid (t, \eta) \xrightarrow{\delta}_D (s, \nu') \mid (t, \eta')}\ [\text{DES-Del}]$$

$$\frac{(s, \nu) \xrightarrow{\mathtt{!a}}_D (s', \nu') \quad (t, \eta) \xrightarrow{\mathtt{?a}}_D (t', \eta')}{(s, \nu) \mid (t, \eta) \xrightarrow{\mathtt{a}}_D (s', \nu') \mid (t', \eta')}\ [\text{DES-}\tau]$$

Figure 7: Semantics of DE-TST (symmetric rules omitted).

Given a DE-TST (X, D), we denote with $\mathrm{uv}(D)$ the set of recursion variables *used* in D, and with $\mathrm{dv}(D)$ the set of recursion variables *defined* in D. We say that (X, D) is *closed* when $\{X\} \cup \mathrm{uv}(D) \subseteq \mathrm{dv}(D)$, and each $X \in \mathrm{uv}(D)$ is defined exactly once. Every TST p can be translated into a DE-TST $\langle p \rangle_V$ (where V is a set of fresh recursion variables, see Definition D.1) in a way that preserves closedness (Lemma D.3) and compliance (Lemma 8.11). Hereafter, we will always assume closed DE-TST.

To define the semantics of DE-TST, we use a new set $\mathcal{S}$ of terms (Definition 8.9). Intuitively, in the state τX the process has performed an internal move (without choosing the branch), while in $[\mathtt{!a}\{g, R\}]\, X$ it has committed to the branch $\mathtt{!a}$.

Definition 8.9 (Semantics of DE-TST). Let $\mathcal{S}$ be a set of terms of the form:

$$t \quad ::= \quad \tau X \ \mid\ [\mathtt{!a}\{g, R\}]\, X \ \mid\ X \qquad\qquad (\text{where } \mathtt{a} \in \mathtt{A},\ g \in \mathcal{G}_{\mathbb{C}}, \text{ and } R \subseteq \mathbb{C})$$

and let D be a set of defining equations. Then, the relation $\to_D$ is inductively defined by the set of rules in Figure 7, where we use $t, s \ldots$ to range over $\mathcal{S}$.

There is almost a one-to-one correspondence between the rules of $\to$ and $\to_D$, aside from the syntactic differences between TST and DE-TST. Rules [DE-Del1] and [DE-Del2] allow time to pass in the same way as [Del] does: they have been split in two only to accommodate with the new term τX. Rule [DE-τ] forces every internal choice to perform an internal step before committing to a branch. However, this is done only if $\nu \in \mathtt{rdy}(p)$, i.e. if at least one of the branches is available. Note that before the internal step has been performed, time cannot pass, hence the only possible move is via [DE-$\oplus$] (so, a sequence of rules [DE-τ] and [DE-$\oplus$] corresponds to rule [$\oplus$] in $\to$).

Definition 8.10 (Compliance for DE-TST). A state $(X, \nu) \mid (Y, \eta)$ in $\to_D$ is *deadlock* whenever (i) it is not the case that both $X \triangleq \mathbf{1}$ and $Y \triangleq \mathbf{1}$ are in D, and (ii) there is no δ and no $\mathtt{a}_\tau \in \mathtt{A} \cup \{\tau\}$ such that $(X, \nu + \delta) \mid (Y, \eta + \delta) \xrightarrow{\mathtt{a}_\tau}_D$. We write $(x, \nu) \bowtie_D (y, \eta)$ when:

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

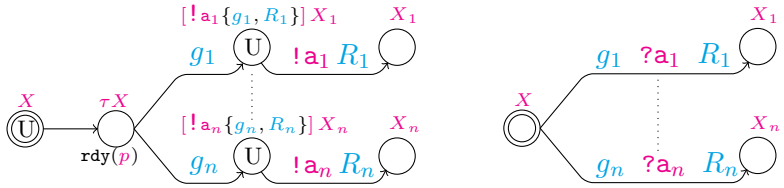

Figure 8: Encoding of an internal choice (left) and of an external choice (right).

and we write $(x, D') \bowtie (y, D'')$ whenever $(x, \nu_0) \bowtie_{D' \cup D''} (y, \eta_0)$.

Lemma 8.11. *Let p and q be two closed TSTs with no shared clocks. Let V_1 and V_2 be two sets of recursion variables not occurring in p and q and such that $V_1 \cap V_2 = \emptyset$. Then:*

$$p \bowtie q \iff \langle p \rangle_{V_1} \bowtie \langle q \rangle_{V_2}$$

Proof. See page 38. □

8.3. **Encoding DE-TST into TA.** In Definition 8.12 we transform DE-TST into TA: first, we use the function $[\![\cdot]\!]$ to transform each defining equation into a TA; then, we compose all the resulting TA with the union operator $\bigsqcup$ (introduced in Definition 8.4). Our encoding produces a location for every term in $\mathcal{S}$ reachable in $\rightarrow_D$; it creates an edge for each move that can be performed by a TST, so that, in the end, the moves performed by the network associated to (X, D) coincide with the moves of X in $\rightarrow_D$. To avoid some technicalities in proofs, we assume that disjunctions never occur in guards (while they can occur in invariants); in Appendix D.1 we discuss how to deal with the general case.

Definition 8.12 (**Encoding of DE-TST into TA**)**.** For all closed DE-TST (X, D), we define the function $\mathcal{T}(X, D) = \bigsqcup_{d \in D}^{X} [\![d]\!]$, where:

$$[\![X \triangleq p]\!] = \begin{cases} \mathit{Idle}(X) & \text{if } p = \mathbf{1} \\ \mathit{Br}(X, \mathtt{rdy}(p), \{(?\mathtt{a}_i, g_i, R_i, \mathit{Idle}(X_i))\}_i) & \text{if } p = \sum_{i \in I} ?\mathtt{a}\{g_i, R_i\}.X_i \\ \mathit{Pfx}(X, \tau, \emptyset, \mathit{Br}(\tau X, \mathtt{rdy}(p), \{(\tau, g_i, \emptyset, A_i)\}_i), \text{where} \\ A_i = \mathit{Pfx}([!\mathtt{a}_i\{g_i, R_i\}]\, X_i, !\mathtt{a}_i, R_i, \mathit{Idle}(X_i)) & \text{if } p = \bigoplus_{i \in I} !\mathtt{a}_i\{g_i, R_i\}.X_i \end{cases}$$

The encoding of $X \triangleq \mathbf{1}$ produces an *idle* TA (Definition 8.5), with a single success location X. The encoding of an *external choice* $X \triangleq \sum_{i \in I} ?\mathtt{a}_i\{g_i, R_i\}.X_i$ generates a location X (with `true` invariant), and outgoing edges towards all the locations X_i: these edges have guards g_i, reset sets R_i, and synchronization labels $?\mathtt{a}_i$ (see Figure 8, right). Basically, the TA is listening on all its channels $?\mathtt{a}_i$, and since the location X is not urgent, time can pass forever while in there. The encoding of $X \triangleq p$ when p is an *internal choice* $\bigoplus_{i \in I} !\mathtt{a}_i\{g_i, R_i\}.X_i$ is a bit more complex (see Figure 8, left). First, we generate an urgent location X, and an edge leading to a non-urgent location τX with invariant $\mathtt{rdy}(p)$. Note that, although $\mathtt{rdy}(p)$ is a semantic object (a set of clock valuations), it can always be represented syntactically as a guard [30]. Second, we connect the latter location to i urgent locations (named $[!\mathtt{a}_i\{g_i, R_i\}]\, X_i$) via internal edges with guards g_i; each of these locations is

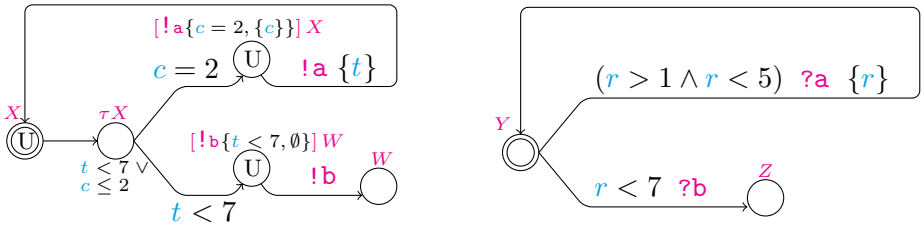

Figure 9: Encoding of the TSTs in Example 8.13.

connected to X_i through an edge with reset set R_i and label $\texttt{!a}_i$. The resulting TA can wait some time before deciding on which branch committing, since time can pass in location τX; however, time passing cannot make all the guards on the branches unsatisfiable, because the invariant $\texttt{rdy}(p)$ on τX ensures that the location is left on time. As soon as this happens, since the arriving location is urgent, time cannot pass anymore, and a synchronization is performed (if possible). The reason we use both locations X and τX is that, in some executions, all the guards of an internal choice may have *already* expired. In this case, the invariant $\texttt{rdy}(p)$ on location τX would be false, and the system could not enter it, so preventing a previous action (if any) to be done. To avoid this problem, we have put an extra location (X) before τX.

Since location names are in $\mathcal{S}$, every defined/used variable X has a location called X, in *every* TA which defines/uses it. When the TA obtained from different equations are composed with $\bigsqcup$, locations with the same name collapse, therefore connecting together the call of a recursion variable with its definition.

Example 8.13. Let $p_1 = \operatorname{rec} X_1.\,(\texttt{!a}\{c = 2, \{c\}\}.X_1 \oplus \texttt{!b}\{t < 7\})$, and let V_1 be a set of recursion variables not in p_1. The DE-TST of p_1 is $\langle p_1 \rangle_{V_1} = (X, D)$, where:

$$D \;=\; \{\; X \triangleq p,\;\; W \triangleq \mathbf{1} \;\} \qquad\qquad \text{where } p = \texttt{!a}\{c = 2, \{c\}\}.X \oplus \texttt{!b}\{t < 7\}.W$$

Figure 9 (left) shows the TA $\mathcal{T}(X, D)$. All the locations but τX and W are urgent; all the invariants are $\texttt{true}$, except for $I(\tau X) = \texttt{rdy}(p) = {\downarrow}([\![c = 2]\!] \cup [\![t < 7]\!])$ (represented by the guard $c \leq 2 \vee t < 7$).

Now, let $q_1 = \operatorname{rec} Y_1.\,\texttt{?a}\{r > 1 \wedge r < 5, \{r\}\}.Y_1 + \texttt{?b}\{r < 7\}$, and let V_2 be a set of recursion variables not in q_1. The DE-TST of q_1 is $\langle q_1 \rangle_{V_2} = (Y, F)$, where:

$$F \;=\; \{\; Y \triangleq q,\;\; Z \triangleq \mathbf{1} \;\} \qquad\qquad \text{where } q = \texttt{?a}\{r > 1 \wedge r < 5, \{r\}\}.Y + \texttt{?b}\{r < 7\}.Z$$

Figure 9 (right) shows the TA $\mathcal{T}(Y, F)$. Its locations are Y and Z (both non-urgent), with invariants $I(Z) = \texttt{true}$ and $I(Y) = \texttt{rdy}(q) = \mathbb{V}$ (represented by the guard $\texttt{true}$).

Lemma 8.14 shows a strict correspondence between the timed LTSs $\to_D$ and $\to_N$: matching states are strongly bisimilar. To make explicit that some state q belongs to a LTS $\to$, we write it as a pair $(q, \to)$.

Lemma 8.14. *Let (X, D') and (Y, D'') be DE-TST such that $\operatorname{dv}(D') \cap \operatorname{dv}(D'') = \emptyset$. Let $N = \mathcal{T}(X, D') \mid \mathcal{T}(Y, D'')$. Then:*

$$((X, \nu_0) \mid (Y, \eta_0), \to_{D' \cup D''}) \sim ((X, Y, \nu_0 \sqcup \eta_0), \to_N)$$

Proof. See page 38. □

8.4. **Decidability of compliance.** To prove that compliance between TSTs is decidable, we reduce such problem to that of checking if a network of TA is deadlock-free — which is known to be decidable [2].

Theorem 8.15. *Let p and q be two closed TSTs. Let V_1 and V_2 be two sets of recursion variables not occurring in p and q and such that $V_1 \cap V_2 = \emptyset$. Then:*

$$p \bowtie q \iff \mathcal{T}\langle p \rangle_{V_1} \mid \mathcal{T}\langle q \rangle_{V_2} \text{ is deadlock-free}$$

Proof. Let $(X, D') = \langle p \rangle_{V_1}$ and $(Y, D'') = \langle q \rangle_{V_2}$. We show that:

$$(X, D') \bowtie (Y, D'') \iff \mathcal{T}(X, D') \mid \mathcal{T}(Y, D'') \text{ is deadlock-free} \tag{8.1}$$

For the ($\Rightarrow$) direction, assume by contradiction that $(X, D') \bowtie (Y, D'')$ but the associated network N is *not* deadlock-free. By Definition 8.3, there exist a reachable deadlocked state s, i.e. $(X, Y, \nu_0 \sqcup \eta_0) \to_N^* s = (x, y, \nu \sqcup \eta)$ and there are no $\delta \geq 0$ and $\mathtt{a}_\tau \in \mathtt{A} \cup \{\tau\}$ such that $s \xrightarrow{\delta}_N \xrightarrow{\mathtt{a}_\tau}_N$. By Lemma 8.14, $(X, \nu_0) \mid (Y, \eta_0) \to_{D' \cup D''}^* (x, \nu) \mid (y, \eta)$, and the last state is bisimilar to s. However, since $(X, D') \bowtie (Y, D'')$, the state $(x, \nu) \mid (y, \eta)$ is not deadlock according to Definition 8.10. Here we have two cases. (i) If $x = \mathbf{1}$ and $y = \mathbf{1}$, then by Definition 8.12 x and y are success locations — contradiction, because s is not success; (ii) $(x, \nu + \delta) \mid (y, \eta + \delta) \xrightarrow{\mathtt{a}_\tau}_{D' \cup D''}$ for some δ and no $\mathtt{a}_\tau \in \mathtt{A} \cup \{\tau\}$. Then, by Lemma 8.14 also s can fire that moves — contradiction. The direction ($\Leftarrow$) is similar. The main statement follows from (8.1) and from Lemma 8.11. □

Example 8.16. Let us consider the two TSTs in Example 8.13. Since the associated network of TA (Figure 9) is deadlock-free, by Theorem 8.15 we conclude that $p_1 \bowtie q_1$.

Our implementation of TSTs (`co2.unica.it`) uses *Uppaal* [15] to check compliance. *Uppaal* can verify deadlock-freedom of a network of TA through its query language, which is a simplified version of Time Computation Tree Logic. In *Uppaal*, the special state formula `deadlock` is satisfied by all deadlocked states; hence, a network is deadlock-free if none of its reachable states satisfies `deadlock`. Note that checking deadlock-freedom with the path formula `A[] not deadlock` would not be correct, because Definition 8.3 does not consider as deadlock the success states. The actual *Uppaal* query we use takes into account success states. E.g., let $N = A_1 \mid A_2$ be a network of TA, with l_1 success location of A_1, and l_2, l_3 success locations of A_2. The query `A[] deadlock imply (A1.l1 && (A2.l2 || A2.l3))` correctly checks deadlock freedom according to Definition 8.3.

9. Runtime monitoring of TSTs

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

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

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

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

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

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

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

Figure 10: Monitoring semantics (symmetric rules omitted).

the monitor outlined above assumes an asymmetry between internal and external choices. Second, in the semantics in Figure 1 all the transitions (both messages and delays) must be allowed by the TSTs: for instance, $(\texttt{!a}\{t \leq 1\}, \nu)$ cannot take any transitions (neither $\texttt{!a}$ 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 9.1 (**Monitoring semantics of TSTs**). *Monitoring configurations* $\gamma, \gamma', \ldots$ are terms of the form $P \parallel Q$, where P and Q are triples (p, c, ν), such that (i) p is either a TST or $\mathbf{0}$, and (ii) 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 $\lambda, \lambda', \ldots \in (\{\mathsf{A}, \mathsf{B}\} \times \mathsf{L}) \cup \mathbb{R}_{\geq 0}$, are defined in Figure 10.

In the rules in Figure 10, we always assume that the leftmost TST is governed by A, while the rightmost one is governed by B. In rule [M-$\oplus$], A has an internal choice, and

she can fire one of her outputs $\texttt{!a}$, provided that its buffer is empty, and the guard g is satisfied. When this happens, the message $\texttt{!a}$ is written to the buffer, and the clocks in R are reset. Then, B can read the buffer, by firing $\texttt{?a}$ in an external choice through rule [M-+]; this requires that the buffer of B is empty, and the guard g of the branch $\texttt{?a}$ 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.

The following lemma establishes that the monitoring semantics is deterministic, i.e., if $\gamma \xrightarrow{\lambda}\!\!\!\!\twoheadrightarrow_M \gamma'$ and $\gamma \xrightarrow{\lambda}\!\!\!\!\twoheadrightarrow_M \gamma''$, then $\gamma' = \gamma''$. This 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. Further, for all finite timed traces $\vec{\lambda}$ (i.e., sequences of actions $\mathsf{A} : \ell$ or time delays δ), there exists some configuration γ reachable from the initial one.

Lemma 9.2. *Let $\gamma_0 = (p, [], \nu_0) \parallel (q, [], \nu_0)$, where $p \bowtie q$. Then, for all finite timed traces $\vec{\lambda}$ there exists one and only one γ such that $\gamma_0 \xrightarrow{\vec{\lambda}}\!\!\!\!\twoheadrightarrow_M \gamma$.*

Proof. Simple inspection of the rules in Figure 10. □

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 9.3 (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 and c is empty, or d is not empty.

Lemma 9.4 states that, in each reachable configuration, only one participant can be on duty; and if no one is on duty nor culpable, then both participants have reached success.

Lemma 9.4. *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.*

Proof. Straightforward by Definition 9.3, Lemma 9.2, and by the rules in Figure 10. □

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

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

and in the final configuration both participants are culpable.

Example 9.5. Let $p = \texttt{!a}\{2 < t < 4\}$ and $q = \texttt{?a}\{2 < t < 5\} + \texttt{?b}\{2 < t < 5\}$ be the TSTs of participants A and B, respectively. Participant A declares that she will send $\texttt{a}$ between 2 and 4 time unit (abbr. t.u.), while B declares that he is willing to receive $\texttt{a}$ or $\texttt{b}$ if they are sent within 2 and 5 t.u. 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 $\vec{\lambda} = \langle 1.2,\ \mathsf{A} : \mathtt{!a},\ \mathsf{B} : \mathtt{?a} \rangle$. Indeed, $\gamma_0 \xrightarrow{\vec{\lambda}}\!\!\!\!\rightarrow_M (\mathbf{1}, [], \nu_0) \parallel (\mathbf{1}, [], \nu_0)$. On the contrary, things may go awry in three cases:

(1) 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.

(2) 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], $\gamma_0 \xrightarrow{6}\!\!\!\!\rightarrow_M (\mathbf{0}, [], \nu_0 + 6) \parallel (q, [], \eta_0 + 6)$, where A is culpable.

(3) 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.

To relate the monitoring semantics in Figure 10 with the one in Figure 1 we use the notion *turn-simulation* of [10]. This relation is between states of two arbitrary 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 9.6 (**Turn-simulation** [10])**.** 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:

(1) $s_1 \rightarrow_1 s_1' \implies \exists s_2' : s_2 \rightarrow_2^* s_2'$ and $s_1'\ \mathcal{R}\ s_2'$

(2) $s_2 \rightarrow_2 s_2' \implies s_1 \rightarrow_1$ or ($s_1\ \mathcal{R}\ s_2'$ and $s_2' \not\rightarrow_2$)

(3) $s_2 \in S_2 \implies s_1 \in S_1$

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 establishes that the two semantics of TSTs (Definitions 2.7 and 9.1) are turn-bisimilar.

Lemma 9.7. *$((p, \nu) \mid (q, \eta), \rightarrow)$ is turn-bisimilar to $((p, [], \nu) \parallel (q, [], \eta), \twoheadrightarrow)$.*

Proof. See page 43. □

When both participants behave honestly, i.e., they never take [M-Fail*] moves, the monitoring semantics preserves compliance (Theorem 9.9). The *monitoring compliance* relation $\bowtie_M$ is the straightforward adaptation of that in Definition 3.1, except that $\twoheadrightarrow$ transitions are used instead of $\rightarrow$ ones (Definition 9.8).

Definition 9.8 (**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:

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

We write $p \bowtie_M q$ whenever $(p, [], \nu_0) \bowtie_M (q, [], \eta_0)$.

Theorem 9.9. $\bowtie\ = \bowtie_M$.

Proof. For the inclusion $\subseteq$, assume that $p \bowtie q$. By Lemma 9.7, the states $(p, \nu_0) \mid (q, \eta_0)$ and $(p, [], \nu_0) \parallel (q, [], \eta_0)$ are turn-bisimilar. By $(p, [], \nu_0) \parallel (q, [], \eta_0) \preccurlyeq (p, \nu_0) \mid (q, \eta_0)$ and $(p, \nu_0) \bowtie (q, \nu_0)$, Lemma 5.9 in [10] implies that $(p, [], \nu_0) \bowtie_M (q, [], \eta_0)$, hence $p \bowtie_M q$. The other inclusion is similar. □

10. Conclusions and Related work

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 computable canonical compliant construction, a decidable subtyping relation, and a decidable runtime monitoring of interactions based on TSTs.

The current article corresponds to a thoroughly revised and improved version of [8]. Besides presenting the proofs of all statements, the current work extends [8] in two main directions. First, in Section 5 it presents an alternative set of kinding rules, which we exploit to show the decidability of kind inference, and from this the computability of the canonical compliant construction. Second, in Section 8 it provides a compliance-preserving encoding of TSTs into timed automata, which we exploit in our decision procedure for compliance.

The work [4] is a step-by-step tutorial on how to exploit timed session types for programming contract-oriented distributed applications.

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 [6]. 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 [18]. 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 [14], [18] 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 [18] external choices can also delay the reading time. The notion of correct interaction studied in [18] 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 [35] 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 addressed in [18]. A possible way to deal with this problem would be to adapt our kind system (in particular, rule [T-+]).

Another problem not addressed by [18] 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 [26]).

Note that our notion of compliance does not imply progress with the semantics of [18] (adapted to the binary case). For instance, consider the TSTs:

$$p \;=\; \mathtt{?a}\{x \le 2\}.\,\mathtt{!a}\{x \le 1\} \qquad\qquad q \;=\; \mathtt{!a}\{y \le 1\}.\,\mathtt{?a}\{y \le 1\}$$

These TSTs are compliant according to Definition 3.1, while using the semantics of [18]:

$$(\nu_0, (p, q, \vec{w_0})) \;\to^*\; (\nu, (\mathtt{!a}\{x \le 1\}, \mathtt{?a}\{y \le 1\}, \vec{w_0})) \qquad \text{with } \nu(x) = \nu(y) > 1$$

This is a deadlocked state, hence p and q do not enjoy progress according to [18].

Dynamic verification of timed multiparty session types is addressed by [39], where the top-down approach to service composition is pursued [32]. Our middleware instead composes and monitors services in a bottom-up fashion [14].

The work [17] studies *communicating timed automata*, a timed version of communicating finite-state machines [20]. In this model, participants in a network communicate asynchronously through bi-directional FIFO channels; similarly to [18], clocks, guards and resets are used to impose time constraints on when communications can happen. A system enjoys *progress* when no deadlocked state is reachable, as well as no orphan messages, unsuccessful receptions, and unfeasible configurations. Since deadlock-freedom is undecidable in the untimed case, then *a fortiori* progress is undecidable for communicating timed automata. So, the authors propose an approximated (sound, but not complete) decidable technique to check when a system enjoys progress. This technique is based on *multiparty compatibility*, a condition which guarantees deadlock-freedom of untimed systems [37].

A classical property of timed systems addressed by [17] is *zenoness*, i.e. the situation in which all the paths from a reachable state are infinite and time-convergent. Again, multiparty compatibility is exploited by [17] to devise an approximated decidable technique which guarantees non-zenoness of communicating timed automata.

In our setting, an example of zenoness is given by the following TSTs:

$$p \;=\; \mathrm{rec}\, X.\,\mathtt{!a}\{x \le 1\}.\,X \qquad\qquad q \;=\; \mathrm{rec}\, X.\,\mathtt{?a}\{x \le 1, x\}.\,X$$

Although $p \bowtie q$, the total elapsed time cannot exceed 1. This implies that, in order to respect p, a participant should have to perform *infinitely* many writings in *a single* time unit. This problem can be solved imposing some restrictions to TSTs, in order to have the property that composition of TSTs is always non-zeno. For instance, this is guaranteed by the notion of *strong non-zenoness* of [43], which can be computed efficiently but is not complete. Another possibility is to check non-zenoness directly in the network of timed automata constructed for checking compliance, using one of the techniques appeared in the literature [30, 43, 19].

In [24] 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 [25]: 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 [24], closed binary systems in TSTs), compatibility/refinement seem unrelated to our notions of compliance/subtyping. Despite the main notions in [24] 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 our main properties, 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.

Acknowledgement

This work has been partially supported by Aut. Reg. of Sardinia P.I.A. 2013 "NOMAD", by EU COST Action IC1201 "Behavioural Types for Reliable Large-Scale Software Systems" (BETTY). We thank the anonymous reviewers for their insightful comments.

References

[1] PayPal buyer protection. https://www.paypal.com/us/webapps/mpp/ua/useragreement-full#13. Accessed: December 31, 2015.

[2] R. Alur and D. L. Dill. A theory of timed automata. *Theor. Comput. Sci.*, 126(2):183–235, 1994.

[3] D. Ancona, V. Bono, M. Bravetti, J. Campos, G. Castagna, P. Deniélou, S. J. Gay, N. Gesbert, E. Giachino, R. Hu, E. B. Johnsen, F. Martins, V. Mascardi, F. Montesi, R. Neykova, N. Ng, L. Padovani, V. T. Vasconcelos, and N. Yoshida. Behavioral types in programming languages. *Foundations and Trends in Programming Languages*, 3(2-3):95–230, 2016.

[4] N. Atzei, M. Bartoletti, T. Cimoli, S. Lande, M. Murgia, A. S. Podda, and L. Pompianu. Contract-oriented programming with timed session types. In *Behavioural Types: from Theory to Tools*. River Publishers, 2017.

[5] F. Barbanera and U. de'Liguoro. Two notions of sub-behaviour for session-based client/server systems. In *PPDP*, pages 155–164. ACM, 2010.

[6] F. Barbanera and U. de'Liguoro. Sub-behaviour relations for session-based client/server systems. *Mathematical Structures in Computer Science*, 25(6):1339–1381, 2015.

[7] M. Bartoletti, I. Castellani, P. Deniélou, M. Dezani-Ciancaglini, S. Ghilezan, J. Pantovic, J. A. Pérez, P. Thiemann, B. Toninho, and H. T. Vieira. Combining behavioural types with security analysis. *J. Log. Algebr. Meth. Program.*, 84(6):763–780, 2015.

[8] M. Bartoletti, T. Cimoli, M. Murgia, A. S. Podda, and L. Pompianu. Compliance and subtyping in timed session types. In *Proc. FORTE*, LNCS, pages 161–177. Springer, 2015.

[9] M. Bartoletti, T. Cimoli, M. Murgia, A. S. Podda, and L. Pompianu. A contract-oriented middleware. In *FACS*, volume 9539 of *LNCS*, pages 86–104. Springer, 2015. http://co2.unica.it.

[10] M. Bartoletti, T. Cimoli, G. M. Pinna, and R. Zunino. Contracts as games on event structures. *J. Log. Algebr. Meth. Program.*, 85(3):399–424, 2016.

[11] M. Bartoletti, T. Cimoli, and R. Zunino. Compliance in behavioural contracts: a brief survey. In *Programming Languages with Applications to Biology and Security*, volume 9465 of *LNCS*, pages 103–121. Springer, 2015.

[12] M. Bartoletti, A. Scalas, E. Tuosto, and R. Zunino. Honesty by typing. *Logical Methods in Computer Science*, 12(4), 2016.

[13] M. Bartoletti, A. Scalas, and R. Zunino. A semantic deconstruction of session types. In *Proc. CONCUR*, volume 8704 of *LNCS*, pages 402–418. Springer, 2014.

[14] M. Bartoletti, E. Tuosto, and R. Zunino. Contract-oriented computing in CO_2. *Sci. Ann. Comp. Sci.*, 22(1), 2012.

[15] G. Behrmann, A. David, and K. G. Larsen. A tutorial on Uppaal. In *Formal methods for the design of real-time systems*, volume 3185 of *LNCS*, pages 200–236. Springer, 2004.
[16] J. Bengtsson and W. Yi. Timed automata: Semantics, algorithms and tools. In *Lectures on Concurrency and Petri Nets, Advances in Petri Nets (ACPN)*, volume 3098 of *LNCS*, pages 87–124. Springer, 2003.
[17] L. Bocchi, J. Lange, and N. Yoshida. Meeting deadlines together. In *CONCUR*, volume 42 of *LIPIcs*, pages 283–296. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2015.
[18] L. Bocchi, W. Yang, and N. Yoshida. Timed multiparty session types. In *CONCUR*, volume 8704 of *LNCS*, pages 419–434. Springer, 2014.
[19] H. Bowman and R. Gómez. How to stop time stopping. *Formal Asp. Comput.*, 18(4):459–493, 2006.
[20] D. Brand and P. Zafiropulo. On communicating finite-state machines. *J. ACM*, 30(2):323–342, 1983.
[21] G. Castagna, M. Dezani-Ciancaglini, E. Giachino, and L. Padovani. Foundations of session types. In *PPDP*, pages 219–230. ACM, 2009.
[22] G. Castagna, N. Gesbert, and L. Padovani. A theory of contracts for web services. *ACM Transactions on Programming Languages and Systems*, 31(5), 2009.
[23] 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.
[24] A. David, K. G. Larsen, A. Legay, U. Nyman, L. Traonouez, and A. Wasowski. Real-time specifications. *STTT*, 17(1):17–45, 2015.
[25] L. de Alfaro and T. A. Henzinger. Interface automata. In *ACM SIGSOFT*, pages 109–120. ACM, 2001.
[26] P.-M. Deniélou and N. Yoshida. Multiparty compatibility in communicating automata: Characterisation and synthesis of global session types. In *ICALP*, volume 7966 of *LNCS*, pages 174–186. Springer, 2013.
[27] M. Dezani-Ciancaglini and U. de'Liguoro. Sessions and session types: An overview. In *WS-FM*, volume 6194 of *LNCS*, pages 1–28. Springer, 2009.
[28] J. S. Dong, P. Hao, S. Qin, J. Sun, and W. Yi. Timed automata patterns. *IEEE Trans. Software Eng.*, 34(6):844–859, 2008.
[29] S. J. Gay and V. T. Vasconcelos. Linear type theory for asynchronous session types. *J. Funct. Program.*, 20(1):19–50, 2010.
[30] T. A. Henzinger, X. Nicollin, J. Sifakis, and S. Yovine. Symbolic model checking for real-time systems. *Inf. Comput.*, 111(2):193–244, 1994.
[31] K. Honda, V. T. Vasconcelos, and M. Kubo. Language primitives and type discipline for structured communication-based programming. In *ESOP*, volume 1381 of *LNCS*, pages 122–138. Springer, 1998.
[32] K. Honda, N. Yoshida, and M. Carbone. Multiparty asynchronous session types. In *POPL*, pages 273–284. ACM, 2008.
[33] K. Honda, N. Yoshida, and M. Carbone. Multiparty asynchronous session types. *J. ACM*, 63(1):9:1–9:67, 2016.
[34] H. Hüttel, I. Lanese, V. T. Vasconcelos, L. Caires, M. Carbone, P. Deniélou, D. Mostrous, L. Padovani, A. Ravara, E. Tuosto, H. T. Vieira, and G. Zavattaro. Foundations of session types and behavioural contracts. *ACM Comput. Surv.*, 49(1):3:1–3:36, 2016.
[35] P. Krcál and W. Yi. Communicating timed automata: The more synchronous, the more difficult to verify. In *CAV*, volume 4144 of *LNCS*, pages 249–262. Springer, 2006.
[36] C. Laneve and L. Padovani. The *must* preorder revisited. In *CONCUR*, volume 4703 of *LNCS*, pages 212–225. Springer, 2007.
[37] J. Lange, E. Tuosto, and N. Yoshida. From communicating machines to graphical choreographies. In *POPL*, pages 221–232. ACM, 2015.
[38] R. Milner. *Communication and concurrency*. Prentice Hall, 1989.
[39] R. Neykova, L. Bocchi, and N. Yoshida. Timed runtime monitoring for multiparty conversations. In *BEAT*, volume 162 of *EPTCS*, pages 19–26, 2014.
[40] D. Sangiorgi. *Introduction to bisimulation and coinduction*. Cambridge University Press, 2012.
[41] K. Takeuchi, K. Honda, and M. Kubo. An interaction-based language and its typing system. In *PARLE*, volume 817 of *LNCS*, pages 398–413. Springer, 1994.
[42] A. Tarski. A lattice-theoretical fixpoint theorem and its applications. *Pacific J. Math.*, 5(2):285–309, 1955.
[43] S. Tripakis. Verifying progress in timed systems. In *AMAST Workshop on Formal Methods for Real-Time and Probabilistic Systems*, volume 1601 of *LNCS*, pages 299–314. Springer, 1999.

[44] N. Yoshida, R. Hu, R. Neykova, and N. Ng. The Scribble protocol language. In *TGC*, volume 8358 of *LNCS*, pages 22–41. Springer, 2013.

Appendix A. Proofs for Section 3

Lemma A.1. *For all* $(p,\nu),(q,\eta)$ *such that* $(p,\nu)\bowtie(q,\eta)$*:*

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

Proof. Trivial. □

Lemma A.2. *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. Assume that $(p,\nu)\,\mathcal{R}\,(q,\eta)$. We proceed by cases on the form of p, modulo unfolding of recursion (note that $\mathcal{R}$ does not talk about committed choices, i.e. terms of the form $[!\mathtt{a}\{g,R\}]\,p$). If $p=\mathbf{1}$, then by Definition 3.5 we have $q=\mathbf{1}$, and so all items are trivial. If p is an internal choice, i.e. $p=\bigoplus_{i\in I}\,!\mathtt{a}_i\{g_i,T_i\}.\,p_i$, by Definition 3.5 we have that $q=\sum_{j\in J}\,?\mathtt{a}_j\{g_j,T_j\}.\,q_j$. Take some δ and i such that $\nu+\delta\in[\![g_i]\!]$ (their existence is guaranteed by Definition 3.5). Let $\nu'=\nu+\delta$ and $\eta'=\eta+\delta$. Then:

$$\dfrac{\dfrac{\nu'\in[\![g_i]\!]}{(\bigoplus_{i\in I}\,!\mathtt{a}_i\{g_i,T_i\}.\,p_i,\nu')\xrightarrow{\tau}([!\mathtt{a}_i\{g_i,T_i\}]\,p_i,\nu')}\,[\oplus]}{(\bigoplus_{i\in I}\,!\mathtt{a}_i\{g_i,T_i\}.\,p_i,\nu')\mid(q,\eta')\xrightarrow{\tau}([!\mathtt{a}_i\{g_i,T_i\}]\,p_i,\;\nu')\mid(q,\eta')}\,[\text{S-}\oplus] \tag{A.1}$$

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 (A.1), with $\delta=0$ and $\gamma=([!\mathtt{a}_i\{g_i,T_i\}]\,p_i,\;\nu)\mid(q,\eta)$. By Definition 3.5, 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 target state is unique, because branch labels in choices are pairwise distinct.

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

$$\dfrac{\dfrac{\nu+\delta\in\mathtt{rdy}(p)=\downarrow\bigcup[\![g_i]\!]}{(p,\nu)\xrightarrow{\delta}(p,\nu+\delta)}\,[\textsc{Del}] \qquad \dfrac{\eta+\delta\in\mathtt{rdy}(q)=\mathbb{V}}{(q,\eta)\xrightarrow{\delta}(q,\eta+\delta)}\,[\textsc{Del}]}{(p,\nu)\mid(q,\eta)\xrightarrow{\delta}(p,\nu+\delta)\mid(q,\eta+\delta)}\,[\text{S-}\textsc{Del}] \tag{A.2}$$

Let $\nu'=\nu+\delta$ and $\eta'=\eta+\delta$. We have to show $(p,\nu')\,\mathcal{R}\,(q,\eta')$. By (A.2) we have that $\nu'\in\mathtt{rdy}(p)$. It remains to show that, whenever $\nu'+\delta'\in[\![g_i]\!]$, there exist j such that $\mathtt{a}_i=\mathtt{a}_j$ and $\eta'+\delta'\in[\![g_j]\!]$ and $(p_i,\nu'+\delta'[T_i])\,\mathcal{R}\,(q_j,\eta'+\delta'[T_j])$. This follows by the assumption $(p,\nu)\,\mathcal{R}\,(q,\eta)$. The case where p is an external choice is similar. □

Proof of Lemma 3.6. 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 ($\Rightarrow$) direction 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 A.1, $q=\sum_{j\in J}\,?\mathtt{a}_j\{g_j,T_j\}\,.\,q_j$. Since $(p,\nu)\mid(q,\eta)$ is not deadlock, it must be $\nu\in\mathtt{rdy}(p)$. By Definition 3.5 it remains to show that, for all δ,i:

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

By contradiction, suppose 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 $\delta=0$, then:

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

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 3.1, 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 ($\Leftarrow$) direction is a straightforward consequence of Lemma A.2 □

Appendix B. Proofs for Section 4

Proof of Theorem 4.4. We have to prove that:

$$\mathrm{fv}(p)\subseteq\mathrm{dom}(\Gamma)\quad\implies\quad\exists!\,\mathcal{K}\ :\ \Gamma\vdash p:\mathcal{K}$$

Let Γ as in the statement. By induction on the structure of p, we have the following cases:

- $p=\mathbf{1}$. Trivial.
- p is a choice (internal or external). Straightforward by the induction hypothesis.
- $p=X$. Since $\mathrm{fv}(p)\subseteq\mathrm{dom}(\Gamma)$, the thesis follows by rule [T-Var].
- $p=\mathrm{rec}\,X\,.\,p'$. Since $\mathrm{fv}(p)\subseteq\mathrm{dom}(\Gamma)$, then for all $\mathcal{K}'$: $\mathrm{fv}(p')\subseteq\mathrm{fv}(p)\cup\{X\}\subseteq\mathrm{dom}(\Gamma)\cup\{X\}=\mathrm{dom}(\Gamma,X:\mathcal{K}')$. Hence, by the induction hypothesis we have that $\exists!\,\mathcal{K}''$: $\Gamma,X:\mathcal{K}'\vdash p':\mathcal{K}''$. The thesis follows by rule [T-Rec]. □

Lemma B.1. *For all $\mathcal{K},\mathcal{K}'$ such that $\mathcal{K}\subseteq\mathcal{K}'$, and for all sets of clocks T:*

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

Proof. Straightforward by Definition 2.6. □

Hereafter, we assume substitutions to be capture avoiding.

Lemma B.2 (Substitution). *Let* $\Gamma \vdash p' : \mathcal{K}'$. *Then, for all* p*:*

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

Proof. We prove that, under the assumption $\Gamma \vdash p' : \mathcal{K}'$, the following items hold:
(1) $\Gamma, X : \mathcal{K}' \vdash p : \mathcal{K} \implies \exists \mathcal{K}'' \supseteq \mathcal{K} : \ \Gamma \vdash p\{p'/X\} : \mathcal{K}''$.
(2) $\Gamma \vdash p\{p'/X\} : \mathcal{K} \implies \exists \mathcal{K}'' \supseteq \mathcal{K} : \ \Gamma, X : \mathcal{K}' \vdash p : \mathcal{K}''$.
Before proving the two items, we show that together they imply the thesis. Assume that $\Gamma, X : \mathcal{K}' \vdash p : \mathcal{K}$. By item 1, $\Gamma \vdash p\{p'/X\} : \mathcal{K}''$ for some $\mathcal{K}'' \supseteq \mathcal{K}$. Then, by item 2, $\Gamma, X : \mathcal{K}' \vdash p : \mathcal{K}'''$ for some $\mathcal{K}''' \supseteq \mathcal{K}''$. Therefore, by uniqueness of kinding, $\mathcal{K}'' \subseteq \mathcal{K}''' = \mathcal{K} \subseteq \mathcal{K}''$. The other direction follows by a similar argument.

To prove item 1, assume $\Gamma, X : \mathcal{K}' \vdash p : \mathcal{K}$. We proceed by induction on the typing rules.

- [T-**1**], [T-Var]. Trivial.
- [T-$\oplus$]. We have:

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

  By the induction hypothesis:

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

  The thesis follows by Lemma B.1.
- [T-$+$]. Similar to [T-$\oplus$].
- [T-Rec]. p must have the form $\mathrm{rec}\, Y \,.\, p''$. If $X = Y$ the thesis is trivial; otherwise:

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

  Since $X \neq Y$, then $p\{p'/X\} = \mathrm{rec}\, Y \,.\, (p''\{p'/X\})$, and for all $\mathcal{K}_0, \mathcal{K}'_0$:

$$\Gamma, X : \mathcal{K}', Y : \mathcal{K}_0 \vdash p' : \mathcal{K}'_0 \iff \Gamma, Y : \mathcal{K}_0, X : \mathcal{K}' \vdash p' : \mathcal{K}'_0 \tag{B.1}$$

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

$$\frac{\exists \mathcal{K}_0, \mathcal{K}'_0 : \ \Gamma, Y : \mathcal{K}_0 \vdash (p''\{p'/X\}) : \mathcal{K}'_0}{\Gamma \vdash p\{p'/X\} : \bigcup \left\{ \tilde{\mathcal{K}} \mid \Gamma, Y : \tilde{\mathcal{K}} \vdash (p''\{p'/X\}) : \tilde{\mathcal{K}}' \wedge \tilde{\mathcal{K}} \subseteq \tilde{\mathcal{K}}' \right\} = \mathcal{K}''} \tag{B.2}$$

  We first show that the premise of (B.2) holds. Since substitutions are capture avoiding, Y is not free in p', and hence $\Gamma, Y : \mathcal{K} \vdash p' : \mathcal{K}'$ as well as Γ. Then, by Equation (B.1) together with the induction hypothesis, the thesis follows. 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:

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

  By (B.1), together with the induction hypothesis, we have that, for some $\tilde{\mathcal{K}}''$:

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

  from which the thesis follows.

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:

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

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

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

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

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

  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:

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

  By (B.1), together with the induction hypothesis, we have that, for some $\tilde{\mathcal{K}}''$:

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

  from which the thesis follows. □

Lemma B.3 (Substitution under dual). *Let $\Gamma \vdash p' : \mathcal{K}$, with X not free in p'. Then, for all p such that p is kindable with environment $\Gamma, X : \mathcal{K}$:*

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

Proof. By induction on the structure of p:

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

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

$$\begin{aligned} &= \sum \mathtt{?a}_i\{g_i, T_i\}\,.\,(\mathrm{co}_{\Gamma, X:\mathcal{K}}(p_i)\,\{^{\mathrm{co}_\Gamma(p')}/_X\}) \\ &= \sum \mathtt{?a}_i\{g_i, T_i\}\,.\,\mathrm{co}_\Gamma(p_i\{^{p'}/_X\}) && \text{by induction hypothesis} \\ &= \mathrm{co}_\Gamma(p\{^{p'}/_X\}) \end{aligned}$$

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

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

$$\begin{aligned} &= \bigoplus \mathtt{!a}_i\{g_i \wedge \mathcal{K}_i[T_i]^{-1}, T_i\}\,.\,(\mathrm{co}_{\Gamma, X:\mathcal{K}}(p_i)\,\{^{\mathrm{co}_\Gamma(p')}/_X\}) \\ &= \bigoplus \mathtt{!a}_i\{g_i \wedge \mathcal{K}_i[T_i]^{-1}, T_i\}\,.\,\mathrm{co}_\Gamma(p_i\{^{p'}/_X\}) && \text{by induction hypothesis} \\ &= \mathrm{co}_\Gamma(p\{^{p'}/_X\}) && \text{by Lemma B.2} \end{aligned}$$

- $p = \mathbf{1}$: Trivial

- $p = Y$: Trivial, whether or not $Y = X$
- $p = \operatorname{rec} Y \,.\, p''$: If $Y = X$ the thesis is trivial. Otherwise, assume $\Gamma, X : \mathcal{K} \vdash \operatorname{rec} Y \,.\, p'' : \mathcal{K}'$. First note that, since we are assuming capture avoiding substitutions, Y must not be free in p'. Indeed, if Y were free in p', then Y would be captured in $p\{{}^{p'}/_{X}\}$. Then:

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

  $$\begin{array}{lll} = & \operatorname{rec} Y \,.\, (\mathrm{co}_{\Gamma, X:\mathcal{K}, Y:\mathcal{K}'}(p'')\,\{{}^{\mathrm{co}_\Gamma(p')}/_{X}\}) & \text{by Definition 4.5} \\ = & \operatorname{rec} Y \,.\, (\mathrm{co}_{\Gamma, Y:\mathcal{K}', X:\mathcal{K}}(p'')\,\{{}^{\mathrm{co}_\Gamma(p')}/_{X}\}) & \text{since } X \neq Y \\ = & \operatorname{rec} Y \,.\, (\mathrm{co}_{\Gamma, Y:\mathcal{K}', X:\mathcal{K}}(p'')\,\{{}^{\mathrm{co}_{\Gamma, Y:\mathcal{K}'}(p')}/_{X}\}) & \text{since } Y \text{ is not free in } p' \\ = & \operatorname{rec} Y \,.\, \mathrm{co}_{\Gamma, Y:\mathcal{K}'}(p''\{{}^{p'}/_{X}\}) & \text{by induction hypothesis} \\ = & \mathrm{co}_{\Gamma, Y:\mathcal{K}'}(p\{{}^{p'}/_{X}\}) & \text{by Definition 4.5} \\ = & \mathrm{co}_\Gamma(p\{{}^{p'}/_{X}\}) & \text{since } Y \text{ is not free in } p\{{}^{p'}/_{X}\} \end{array}$$

Note that the induction hypothesis is trivially applicable: p is kindable in $\Gamma, X : \mathcal{K}$, necessarily by rule [T-Rec]. The premise of rule [T-Rec] implies p'' is kindable in $\Gamma, X : \mathcal{K}, Y : \mathcal{K}'$. □

Definition B.4. We define the structural congruence relation $\equiv$ between TSTs as the least congruence relation closed under the following equation:

$$\operatorname{rec} X \,.\, p \equiv p\{{}^{\operatorname{rec} X \,.\, p}/_{X}\}$$

Lemma B.5. *For all kindable p and q, we have that: $p \equiv q \implies \mathrm{co}(p) \equiv \mathrm{co}(q)$.*

Proof. The thesis follows by the following more general statement:

$$\forall \Gamma \;:\; \forall p, q \text{ kindable in } \Gamma \;:\; p \equiv q \implies \mathrm{co}_\Gamma(p) \equiv \mathrm{co}_\Gamma(q)$$

The proof is by easy induction on the structure of p, using Lemma B.3 in the case $p = \operatorname{rec} X \,.\, p'$, which is applicable because the substitution $p'\{{}^{\operatorname{rec} X \,.\, p'}/_{X}\}$ does not capture the free variables in $\operatorname{rec} X \,.\, p'$. □

Lemma B.6. *Every closed p is structurally equivalent to a TST of the following shape:*

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

Proof. Trivial. □

Proof of Theorem 4.6 (Soundness). Let:

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

By Lemma 3.6, it is enough to show that $\mathcal{R}$ is a coinductive compliance relation (Definition 3.5). Assume that $\Gamma \vdash p : \mathcal{K}$ and $\nu \in \mathcal{K}$. We proceed by cases on the shape of p. According to Lemma B.6, we have the following cases:

- $p \equiv \mathbf{1}$. By Lemma B.5 $\mathrm{co}(p) \equiv \mathbf{1}$, from which the thesis follows.
- $p \equiv \bigoplus \mathtt{!a}_i\{g_i, T_i\} \,.\, p_i$. By Lemma B.5, $\mathrm{co}(p) \equiv \sum \mathtt{?a}_i\{g_i, T_i\} \,.\, \mathrm{co}(p_i)$. By rule [T-⊕] it follows that $\nu \in \mathtt{rdy}(p)$. To conclude:

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

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

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

from which the thesis follows.

- $p \equiv \sum \mathtt{?a}_i\{g_i, T_i\}\,.\,p_i$. By Lemma B.5, $\mathrm{co}(p) = \bigoplus \mathtt{!a}_i\{g_i \wedge \mathcal{K}_i, T_i\}\,.\,\mathrm{co}(p_i)$ with $\vdash p_i : \mathcal{K}_i$. By 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 the thesis follows.

Definition B.7. We define the function Φ from TSTs to sets of clock valuations as follows:
$$\Phi(p) \stackrel{\text{def}}{=} \{\nu \mid \exists q, \eta : (p,\nu) \bowtie (q,\eta)\}$$

Lemma B.8. *Let* $\Gamma = X_1 : \Phi(p_1), \ldots, X_m : \Phi(p_m)$*, where all* p_i *are closed, and let* $\vec{X} = (X_1, \ldots, X_m), \vec{p} = (p_1, \ldots, p_m)$*. Then, for all* p *such that* $\mathrm{fv}(p) \subseteq \vec{X}$*:*
$$\Gamma \vdash p : \mathcal{K} \implies \Phi(p\{\vec{p}/\vec{X}\}) \subseteq \mathcal{K}$$

Proof. We start with an auxiliary definition. The recursion nesting level (RL) for a TST is inductively defined as follows:
$$\mathrm{RL}(\mathbf{1}) \stackrel{\text{def}}{=} \mathrm{RL}(X) \stackrel{\text{def}}{=} 0 \qquad \mathrm{RL}(\textstyle\sum_{i\in I} \mathtt{?a}_i\{g_i, T_i\}\,.\,p_i) \stackrel{\text{def}}{=} max\ \{\mathrm{RL}(p_i)\}_{i\in I}$$
$$\mathrm{RL}(\mathrm{rec}\,X\,.\,p) \stackrel{\text{def}}{=} 1 + \mathrm{RL}(p) \qquad \mathrm{RL}(\textstyle\bigoplus_{i\in I} \mathtt{!a}_i\{g_i, T_i\}\,.\,p_i) \stackrel{\text{def}}{=} max\ \{\mathrm{RL}(p_i)\}_{i\in I}$$
We then define the relation $\prec$ as:
$$p \prec q \text{ whenever } \mathrm{RL}(p) < \mathrm{RL}(q) \vee (\mathrm{RL}(p) = \mathrm{RL}(q) \wedge p \text{ is a strict subterm of } q)$$
It is trivial to check that $\prec$ is a well-founded relation (exploiting the fact that the strict subterm relation is well-founded as well). We then proceed by well-founded induction on $\prec$. 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 $i \in \{1, \ldots, m\}$. $\mathcal{K} = \Gamma(X_i) = \Phi(p_i) = \Phi(X_i\{\vec{p}/\vec{X}\})$.
- $p = \sum_{i\in I} \mathtt{?a}_i\{g_i, T_i\}\,.\,p_i$.
$$\frac{\Gamma \vdash p_i : \mathcal{K}_i \qquad \text{for } i \in I}{\Gamma \vdash \sum_{i\in I} \mathtt{?a}_i\{g_i, T_i\}\,.\,p_i : \bigcup_{i\in I} \downarrow \left([\![g_i]\!] \cap \mathcal{K}_i[T_i]^{-1}\right) = \mathcal{K}}\ [\text{T-+}]$$
Since, for all $i \in I$ it holds $p_i \prec p$, by the induction hypothesis:
$$\Phi(p_i\{\vec{p}/\vec{X}\}) \subseteq \mathcal{K}_i \tag{B.3}$$
Now suppose, by contradiction, $\Phi(p\{\vec{p}/\vec{X}\}) \not\subseteq \mathcal{K}$. Then there exist ν, η, q such that $\Phi(p\{\vec{p}/\vec{X}\}) \ni \nu \notin \mathcal{K}$ and $(p\{\vec{p}/\vec{X}\}, \nu) \bowtie (q, \eta)$. By Definition 3.5 we have $q = \bigoplus_{j\in J} \mathtt{!a}_j\{g_j, T_j\}\,.\,q_j$, with $\eta \in \mathtt{rdy}(q)$ and
$$\forall \delta, j : \ \eta + \delta \in [\![g_j]\!] \implies$$
$$\exists i : \mathtt{a}_i = \mathtt{a}_j \wedge \nu + \delta \in [\![g_i]\!] \wedge (p_i\{\vec{p}/\vec{X}\}, (\nu+\delta)[R_i]) \bowtie (q_j, (\eta+\delta)[R_j])$$
But then, by Definition B.7 and Equation (B.3):
$$(\nu+\delta)[R_i] \in \Phi(p_i\{\vec{p}/\vec{X}\}) \subseteq \mathcal{K}_i$$
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.

- $p = \operatorname{rec} Y\,.\,p'$. By rule [T-Rec]:

$$\frac{\Gamma, Y : \mathcal{K}_0 \vdash p' : \mathcal{K}'_0}{\Gamma \vdash \operatorname{rec} Y\,.\,p' : \bigcup \{\mathcal{K} \mid \exists \mathcal{K}' \supseteq \mathcal{K} :\ \Gamma, Y : \mathcal{K} \vdash p' : \mathcal{K}'\} = \mathcal{K}}$$

To prove $\Phi(p\{\vec{p}/\vec{X}\}) \subseteq \mathcal{K}$, it is enough to show that, for some $\mathcal{K}' \supseteq \Phi(p\{\vec{p}/\vec{X}\})$:

$$\Gamma, Y : \Phi(p\{\vec{p}/\vec{X}\}) \vdash p' : \mathcal{K}'$$

Since compliance is preserved by unfolding of recursion (because TSTs are considered up-to $\equiv$ in Definition 2.7), by Definition B.7 it follows that:

$$\Phi(p\{\vec{p}/\vec{X}\}) = \Phi(p'\{\vec{p}/\vec{X}\setminus\{Y\}\}\{^{\operatorname{rec} Y\,.\,(p'\{\vec{p}/\vec{X}\setminus\{Y\}\})}/_{Y}\}) = \Phi(p'\{\vec{p'}/\vec{X'}\})$$

where $\vec{X'}$ and $\vec{p'}$ are defined as follows:

$$\vec{X'} = \begin{cases} \vec{X}Y & \text{if } Y \notin \vec{X} \\ \vec{X} & \text{otherwise} \end{cases} \qquad \vec{p'} = \begin{cases} (p_1, \ldots, p_{i-1}, p\{\vec{p}/\vec{X}\}, p_{i+1}, \ldots, p_m) & \text{if } Y = X_i \\ \vec{p}(p\{\vec{p}/\vec{X}\}) & \text{otherwise} \end{cases}$$

Since $\mathrm{RL}(p') < \mathrm{RL}(p)$, by the induction hypothesis we have:

$$\Gamma, Y : \Phi(p'\{\vec{p'}/\vec{X'}\}) \vdash p' : \mathcal{K}' \implies \Phi(p'\{\vec{p'}/\vec{X'}\}) \subseteq \mathcal{K}'$$

Since $\mathrm{fv}(p) \subseteq \vec{X} = \mathrm{dom}(\Gamma)$ it follows that $\mathrm{fv}(p') \subseteq \vec{X'} = \mathrm{dom}(\Gamma, Y : \Phi(p'\{\vec{p'}/\vec{X'}\}))$, and hence, by Theorem 4.4, the premise of the induction hypothesis is satisfied, and we can conclude $\Phi(p'\{\vec{p'}/\vec{X'}\}) \subseteq \mathcal{K}'$. □

Proof of Theorem 4.8 (Completeness). Suppose $\vdash p : \mathcal{K}$ and $\exists q, \eta.\ (p, \nu) \bowtie (q, \eta)$. By instantiating Lemma B.8 with the empty kinding environment, we obtain $\Phi(p) \subseteq \mathcal{K}$. Hence, by Definition B.7 we conclude that $\nu \in \mathcal{K}$. □

Proof of Lemma 4.9. We prove that the relation:

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

is a coinductive compliance relation (Definition 3.5). We proceed by cases on the form of p:

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

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

with $\vdash p_j : \mathcal{K}_j$ for all $j \in J$ and, by Definition 3.5, for all δ, i such that $\nu + \delta \in [\![g_i]\!]$, there exists j such that $\mathtt{a}_i = \mathtt{a}_j$, $\nu' + \delta \in [\![g_j]\!]$, $(p_i, (\nu + \delta)[T_i]) \bowtie ({p_j}', (\nu' + \delta)[T_j])$, and $\nu + \delta \in \mathtt{rdy}(p)$. Hence, $q = \sum \mathtt{?a}_k\{g_k, T_k\}\,.\,p_k$, and for all δ, j such that $\nu' + \delta \in [\![g_j]\!]$, there exists k such that $\mathtt{a}_k = \mathtt{a}_j$, $\eta + \delta \in [\![g_k]\!]$, $(\mathrm{co}({p_j}'), (\nu' + \delta)[T_j]) \bowtie (q_k, (\eta + \delta)[T_k])$, and $\nu' \in \mathtt{rdy}(\mathrm{co}(p'))$. 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 we should have $(p_i, (\nu + \delta)[T_i]) \bowtie ({p_j}', (\nu' + \delta)[T_j])$ and $(\nu' + \delta)[T_j] \notin \mathcal{K}_j$ — contradiction by Theorem 4.8.

- $p = \sum \mathtt{?a}_i\{g_i, T_i\}\,.\,p_i$. It must be:

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

and for all δ, j such that $\nu' + \delta \in [\![g_j]\!]$, there exists i such that $\mathtt{a}_j = \mathtt{a}_i$, $\nu + \delta \in [\![g_i]\!]$, $({p_j}', \nu' + \delta[T_j]) \bowtie (p_i, \nu + \delta[T_j])$, and $\nu' \in \mathtt{rdy}(p')$. Hence, $q = \bigoplus \mathtt{?a}_k\{g_k, T_k\}\,.\,p_k$, and for all δ, k such that $\eta + \delta \in [\![g_k]\!]$, there exists j such that $\mathtt{a}_k = \mathtt{a}_j$, $\nu' + \delta \in [\![g_j]\!]$, $(q_k, (\eta + \delta)[T_k]) \bowtie ({p_j}', (\nu + \delta)[T_i])$, and $\eta \in \mathtt{rdy}(q)$. The thesis follows by the composition of the above. □

Appendix C. Proofs for Section 5

Theorem C.1 (**Knaster-Tarski fixed point theorem** [42]). *Let $\mathcal{U} = (U, \leq)$ be a complete lattice, and let f be a monotonic endofunction over U. Then $(P, \leq)$, where P is the set of fixed points of f, is a complete lattice. In particular, we have:*

$$\bigsqcup P = \bigsqcup \{x \mid x \leq f(x)\} \qquad \bigsqcap P = \bigsqcap \{x \mid f(x) \leq x\}$$

Definition C.2 (**Cocontinuous function**). An endofunction f over a complete lattice D is said to be cocontinuous iff, for all sequences $d_0, d_1, \ldots$ of decreasing points in D, we have:

$$f(\textstyle\bigsqcap_i d_i) \;=\; \bigsqcap_i f(d_i)$$

Proof of Lemma 5.2. For item 1, let $d_0, d_1, \ldots$ be a decreasing sequence of elements in D. By finiteness of D, there must exist some n such that, for all n', $d_n = d_{n+n'}$. Clearly, $f(\bigsqcap_i d_i) = f(d_n)$. By monotonicity, the sequence $f(d_0), f(d_1), \ldots$ is decreasing, and its meet has to be $f(d_n)$.

For item 2, we show, by induction on n, that, for all $n \geq 0$, $\mathrm{gfp}(f) \leq f^n(\top)$. The base case is trivial, since $f^0(\top) = \top$. For the induction case, suppose that $\mathrm{gfp}(f) \leq f^n(\top)$. By monotonicity of f we have that: $\mathrm{gfp}(f) = f(\mathrm{gfp}(f)) \leq f(f^n(\top)) = f^{n+1}(\top)$.

For item 3, see [40]. □

Proof of Lemma 5.4. We define the partial order $\sqsubseteq$ between environments as follows:

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

and we show that, for all Γ, Γ' such that $\Gamma \sqsubseteq \Gamma'$:

$$\Gamma \vdash p : \mathcal{K} \implies \exists \mathcal{K}' \supseteq \mathcal{K} :\; \Gamma' \vdash p : \mathcal{K}' \tag{C.1}$$

This can be done by easy induction on the structure of p. To prove the main statement, let $\mathrm{fv}(p) \subseteq \mathrm{dom}(\Gamma) \cup \{X\}$, and suppose $\mathcal{K} \subseteq \mathcal{K}'$. We have to show $F(\mathcal{K}) \subseteq F(\mathcal{K}')$. Expanding (5.3): $\Gamma, X : \mathcal{K} \vdash p : \mathcal{K}_0$ and $\Gamma, X : \mathcal{K}' \vdash p : \mathcal{K}_1$, for some $\mathcal{K}_0, \mathcal{K}_1$ such that $\mathcal{K}_0 \subseteq \mathcal{K}_1$. Such $\mathcal{K}_0, \mathcal{K}_1$ exist and are unique (Theorem 4.4); $\mathcal{K}_0 \subseteq \mathcal{K}_1$ follows by (C.1). □

Proof of Lemma 5.6. By structural induction on p. The only non-trivial case is when $p = \operatorname{rec} X\,.\,p'$.

For the ($\Rightarrow$) direction, suppose that:

$$\frac{\Gamma, X : \mathcal{K}'' \vdash p' : \mathcal{K}'''}{\Gamma \vdash \operatorname{rec} X\,.\,p' : \bigcup\{\mathcal{K}_0 \mid \exists \mathcal{K}_1 \supseteq \mathcal{K}_0 :\ \Gamma, X : \mathcal{K}_0 \vdash p' : \mathcal{K}_1\} = \mathcal{K}}\ \text{[T-Rec]}$$

and recall that, by Theorem C.1 and Lemma 5.4, $\mathcal{K} = \operatorname{gfp}(F_{\Gamma,X,p})$. By the induction hypothesis, for all zones $\mathcal{Z}$:

$$F_{\Gamma,X,p'}(\mathcal{Z}) = \hat{F}_{\Gamma,X,p'}(\mathcal{Z}) \tag{C.2}$$

Hence, Lemma 5.4 implies that $\hat{F}_{\Gamma,X,p'}$ is monotonic (on the lattice of zones). Since this lattice is finite, then $\hat{F}_{\Gamma,X,p'}$ is cocontinuous (item 1 of Lemma 5.2). By (C.2) and Theorem 4.4, $\hat{F}^i_{\Gamma,X,p}(\mathcal{K}_0)$ is defined for all $\mathcal{K}_0$ and i. Hence, by rule [I-Rec] and item 3 of Lemma 5.2:

$$\Gamma \vdash_I \operatorname{rec} X\,.\,p' : \textstyle\bigsqcap_{i\geq 0} \hat{F}^i_{\Gamma,X,p'}(\mathbb{V}) \;=\; \hat{\mathcal{K}} \;=\; \operatorname{gfp}(\hat{F}_{\Gamma,X,p'}) \tag{C.3}$$

To conclude that $\mathcal{K} = \hat{\mathcal{K}}$ it is enough to show that $\operatorname{gfp}(F_{\Gamma,X,p'}) = \hat{\mathcal{K}}$. By (C.2) and (C.3) it follows that $\hat{\mathcal{K}}$ is a fixed point of $F_{\Gamma,X,p'}$, and so by definition of gfp, $\hat{\mathcal{K}} \subseteq \operatorname{gfp}(F_{\Gamma,X,p'})$. By (C.2) and item 2 of Lemma 5.2 we conclude that $\operatorname{gfp}(F_{\Gamma,X,p'}) \subseteq \hat{\mathcal{K}}$.

For the ($\Leftarrow$) direction, suppose that by rule [I-Rec] we have:

$$\Gamma \vdash_I \operatorname{rec} X\,.\,p' : \textstyle\bigsqcap_{i\geq 0} \hat{F}^i_{\Gamma,X,p'}(\mathbb{V})$$

Since $\bigsqcap_{i\geq 0} \hat{F}^i_{\Gamma,X,p'}(\mathbb{V})$ is defined, the induction hypothesis gives $\Gamma, X : \mathbb{V} \vdash p' : F_{\Gamma,X,p'}(\mathbb{V})$, hence the premise of rule [T-Rec] is satisfied. We can conclude with the same argument as in the previous case.

Appendix D. Proofs for Section 8

Definition D.1 (DE-TST transformation). Let p be a TST according to Definition 2.2. Let V be an infinite set of recursion variables not occurring in p. Then, the DE-TST of p, denoted by $\langle p \rangle$, is given by :

$$\begin{array}{lll}
(a) & \langle \bigcirc_{i=1}^n \ell_{\mathtt{i}}\{g_i, R_i\}.p_i\rangle \;=\; & (X_0,\ \bigcup_{i=1}^n D_i \cup \{X_0 \triangleq \bigcirc_{i=1}^n \ell_{\mathtt{i}}\{g_i, R_i\}.X_i\}) \\
 & & \quad \text{for } \bigcirc \in \{\oplus, \sum\},\ \text{with } X_0 \in V \text{ and} \\
 & & \quad (X_i, D_i) = \langle p_i \rangle_{V \setminus W_i} \text{ and} \\
 & & \quad W_i = (\{X_0\} \cup \bigcup_{j=1}^{i-1} \operatorname{dv}(D_j)) \\
(b) & \langle \operatorname{rec} X\,.\,p' \rangle \;=\; & (X_0, D[X_0/X]) \quad \text{where } (X_0, D) = \langle p' \rangle \\
(c) & \langle X \rangle \;=\; & (X, \emptyset) \\
(d) & \langle \mathbf{1} \rangle \;=\; & (X_0, \{X_0 = \mathbf{1}\}) \quad \text{where } X_0 \in V
\end{array}$$

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

Example D.2. Consider the following translations of TSTs, where V is a set of recursion variables not occurring in any p_i (we omit guards and reset sets).

$$\begin{aligned}
\langle \mathtt{!a} \oplus \mathtt{!b} \rangle &= (X_0, \{X_0 \triangleq \mathtt{!a}.X_1 \oplus \mathtt{!b}.X_2,\ X_1 \triangleq \mathbf{1},\ X_2 \triangleq \mathbf{1}\}) \\
\langle \mathtt{!a}.\mathrm{rec}\, X.\, (\mathtt{!b}.X \oplus \mathtt{!c}) \rangle &= (X_0, \{X_0 \triangleq \mathtt{!a}.X_1,\ X_1 \triangleq \mathtt{!b}.X_1 \oplus \mathtt{!c}.X_2,\ X_2 \triangleq \mathbf{1}\}) \\
\langle \mathrm{rec}\, X.\, \mathtt{!a}.\mathrm{rec}\, Y.\, (\mathtt{!b}.X \oplus \mathtt{!c}.Y) \rangle &= (X_0, \{X_0 \triangleq \mathtt{!a}.X_1,\ X_1 \triangleq \mathtt{!b}.X_0 \oplus \mathtt{!c}.X_1\}) \\
\langle \mathrm{rec}\, X.\, (\mathtt{!a}.\mathrm{rec}\, X.\, \mathtt{!b}.X \ \oplus\ \mathtt{!c}.X) \rangle &= (X_0, \{X_0 \triangleq \mathtt{!a}.X_1 \oplus \mathtt{!c}.X_0,\ X_1 \triangleq \mathtt{!b}.X_1\})
\end{aligned}$$

Lemma D.3. *If p is closed, then $\langle p \rangle$ is closed.*

Proof. (Sketch). A DE-TST is closed if its used recursion variables plus the initial one are defined exactly once. In the transformation, we see by construction that all the new variable are first declared. The only open issue is caused by variables already presents in the TST. However, if a TST is closed, then all its used variables are in the scope of some rec declaration, and hence they will be correctly renamed by rule D.1(b). □

Proof of Lemma 8.11. *(Sketch).* Rules in $\to$ and $\to_D$ have a strong correspondence: first of all, all the terms in configurations but one are the same, secondly every move but one in one system corresponds to one move in the other. The only exception concerns rule [⊕] in $\to$ which corresponds to pair ([DE-τ], [DE-⊕]) in $\to_D$, and vice-versa. Hence, the proof is done proceeding by absurd: if only one of the systems were deadlock and the other not, the other could still move and since rules and configurations are the same, also could the first one, leading to a contradiction.

Lemma D.4. *Let (X, D) be a DE-TST, and let $A = [\![D]\!]^X = (Loc, Loc^u, l^0, E, I)$. Then:*

$$I(l) = \begin{cases} \mathtt{rdy}(p) & \text{if } l = \tau Y, \text{ for some } Y \text{ and } X \triangleq p \in D \\ \mathit{true} & \text{otherwise} \end{cases} \qquad (\forall l \in Loc)$$

Proof. By Definition 8.4, the invariant of A is given by $I(l) = \bigwedge_i I_i(l)$ for all l, where each I_i is the invariant of the encoding of some defining equation in D. By Definition 8.12, the only location with invariant other than `true` has the form τY, which can only be obtained by encoding $Y \triangleq p$. Since (X, D) is closed, there is exactly one defining equation for Y. Hence, τY occurs only in one TA (say, in A_j), and so $I(l) = \bigwedge_i I_i(l) = I_j(l) = \mathtt{rdy}(p)$. □

Proof of Lemma 8.14. Let (X, D') and (Y, D'') as in the statement, and let $\mathcal{T}(X, D') = (Loc_1, Loc_1^u, l_1^0, E_1, I_1)$ and $\mathcal{T}(Y, D'') = (Loc_2, Loc_2^u, l_2^0, E_2, I_2)$. We show that:

$$\mathcal{R} = \{((x,\nu)|(y,\eta),\quad (x, y, \nu \sqcup \eta)) \mid x, y, z \in \mathcal{S} \text{ and } \nu \in [\![I_1(x)]\!] \text{ and } \eta \in [\![I_2(y)]\!]\} \qquad \text{(D.1)}$$

is a bisimulation. We denote with $\mathtt{ck}(D)$ the set of clocks used in D. First, we show that every possible move from $(x,\nu)|(y,\eta)$ in $\to_D$ is matched by a move of $(x, y, \nu \cup \eta)$ in $\to_N$, and that the resulting states are related by $\mathcal{R}$. We have the following cases, according to the rule used to move:

- [DES-⊕].

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

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

– [DE-τ]. We have that $(x,\nu) \xrightarrow{\tau}_D (\tau X,\nu)$ with $x = X$ and $X \triangleq p \in D$ with $p = \oplus\ldots$ and $\nu \in \mathtt{rdy}(p)$. By Definition 8.12, the mapping of an internal choice is $[\![X \triangleq p]\!] = \mathit{Pfx}(X,\tau,\emptyset,\mathit{Br}(\tau X,\mathtt{rdy}(p),\{(\tau,g_i,\emptyset,A_i)\}_i)$ for some A_i and g_i. Hence, by definition of the *Pfx* pattern, there exists an edge in E_1 such as $(X,\tau,\mathtt{true},\emptyset,\tau X)$. By definition of the *Br* pattern and by Lemma D.4, $I_1(\tau X) = \mathtt{rdy}(p)$. In our case rule [TA1] of Definition 8.2 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 $\nu \sqcup \eta \in [\![\mathtt{true}]\!]$ and $\nu \sqcup \eta \in [\![I_1(x) \wedge I_2(y)]\!]$. The former holds trivially. For the second, according to Lemma D.4 we have $I_1(\tau X) = \mathtt{rdy}(p)$; by premises in [DES-$\oplus$] we have $\nu \in \mathtt{rdy}(p)$ and, since $\mathtt{ck}(D') \cap \mathtt{ck}(D'') = \emptyset$, we have $\nu \sqcup \eta \in \mathtt{rdy}(p)$. $\nu \sqcup \eta \in [\![I_2(y)]\!]$ holds by hypothesis in the definition of $\mathcal{R}$ in Equation (D.1). Hence $(X,y,\nu \sqcup \eta) \xrightarrow{\tau}_N (\tau X,y,\nu \sqcup \eta)$ and the resulting states belong to $\mathcal{R}$.

– [DE-$\oplus$]. We have that $(x,\nu)|(y,\eta) \xrightarrow{\tau}_D (x',\nu)|(y,\eta)$ with $x = \tau X$ and $X \triangleq \mathtt{!a}\{g,R\}.\,Y \oplus p'$ and $\nu \in [\![g]\!]$ and $x' = [\mathtt{!a}\{g,R\}]\,Y$. By Definition 8.12, the mapping of an internal choice is

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

for some set S, and with $A = \mathit{Pfx}([\mathtt{!a}\{g,R\}]\,Y,\mathtt{!a},R,\mathit{Idle}(Y))$. By definition of the *Br* pattern and Lemma D.4, there exists an edge in E_1 such as $(\tau X,\tau,g,\emptyset,x')$ with $I_1(x') = \mathtt{true}$. In our case rule [TA1] of Definition 8.2 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, according to Lemma D.4 we have $I_1(\tau X) = \mathtt{true}$– 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 (D.1). 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.

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

By Definition 8.12 and Lemma D.4, there is only a case in which a location name has prefix τ, and in that case we have $I_1(\tau X) = \mathtt{rdy}(p)$ and $\tau X \notin \mathit{Loc}_1^y$. Similarly, $I_2(\tau Y) = \mathtt{rdy}(q)$ and $\tau Y \notin \mathit{Loc}_2^y$. In our case rule [TA3] of Definition 8.2 states that:

$$\frac{((\nu \sqcup \eta)+\delta) \in [\![I_1(\tau X) \wedge I_2(\tau Y)]\!] \qquad \tau X \notin \mathit{Loc}_1^y \qquad \tau Y \notin \mathit{Loc}_2^y}{(\tau X,\tau Y,\nu \sqcup \eta) \xrightarrow{\delta}_N (\tau X,\tau Y,(\nu \sqcup \eta)+\delta)}$$

By [DE-DEL2] rule, we have $\nu + \delta \in \mathtt{rdy}(p)$, hence, since $\mathtt{ck}(D') \cap \mathtt{ck}(D'') = \emptyset$ and so $(\nu \sqcup \eta) + \delta \in \mathtt{rdy}(p)$. 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].

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

By Definition 8.12 and Lemma D.4, 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 $\tau Y \notin Loc_2^y$. 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 8.12 and Lemma D.4, the mapping for an external choice is: $[\![X \triangleq p]\!] = Br(X, \mathtt{rdy}(p), S)$ 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 $\tau Y \notin Loc_1^y$. In this case rule [TA3] of Definition 8.2 states that:

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

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 $\mathcal{R}$.

In the second case, according with Definition 8.12, the mapping for the success termination is: $[\![X \triangleq \mathbf{1}]\!] = Idle(X)$. By definition of the *Idle* pattern and Lemma D.4, the location X is not urgent, with $I_1(X) = \mathtt{true}$. Again $\tau Y \notin Loc_1^y$. In this case rule [TA3] of Definition 8.2 states that:

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

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].

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

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

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

Let $x = [\mathtt{!a}\{g,R\}]\,X'$. By Definition 8.12, the last part of the mapping for an internal choice is such as $Pfx(x, \mathtt{!a}, R, Idle(X'))$. By definition of the *Pfx* pattern and Lemma D.4, there exists an edge $(x, \mathtt{!a}, \mathtt{true}, R, X') \in E_1$ and $I_1(X') = \mathtt{true}$. By Definition 8.12, the

mapping for an external choice is: $[\![Y \triangleq \texttt{?a}\{f,T\}.Y' + q]\!] = Br(Y, \texttt{true}, \{(\texttt{?a}, f, T, Idle(Y'))\} \cup S)$ for some S. By definition of the Br pattern and Lemma D.4, there exists an edge $(Y, \texttt{?a}, f, T, Y')$ and $I_1(Y') = \texttt{true}$. In this case rule [TA2] of Definition 8.2 states that:

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

Since by hypothesis $\eta \in [\![f]\!]$, since $\texttt{ck}(D') \cap \texttt{ck}(D'') = \emptyset$ and we obtain $(\eta \sqcup \nu) \in [\![f]\!]$. Hence by rule [TA2] of Definition 8.2 , we have $(x, Y, \nu \sqcup \eta) \xrightarrow{\texttt{a}}_N (X', Y', \nu \sqcup \eta[R][T]$. The resulting states, belong to $\mathcal{R}$.

We now show that every possible move of $(x, y, \nu \sqcup \eta)$ in $\rightarrow_N$ is matched by a move of $(x, \nu)|(y, \eta)$ in $\rightarrow_D$, and that the resulting states are related by $\mathcal{R}$. We have the following cases, according to the rules of Definition 8.2 used to justify the move:

- [TA1].

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

  By Definition 8.12, 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 $[\![X \triangleq p]\!] = Pfx(X, \tau, \emptyset, Br(\tau X, \texttt{rdy}(p), \{(\tau, g, \emptyset, A)\} \cup S))$ for some set S and TA A. By definition of the Pfx and Br patterns and by Lemma D.4, 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) Assume that we fire the first edge $(X, \tau, \texttt{true}, \emptyset, \tau X)$. Hence $x = X$, $x' = \tau X$ and $R = \emptyset$. By hypothesis $\nu \sqcup \eta \in [\![I_1(\tau X)]\!] = \texttt{rdy}(p)$. Since $\texttt{ck}(D') \cap \texttt{ck}(D'') = \emptyset$, we derive $\nu \in \texttt{rdy}(p)$. Hence, we can use [DE-τ] and [DES-$\oplus$] rule from Definition 8.9 to obtain:

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

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

  (ii) Assume that we fire the second edge $(\tau X, \tau, g, \emptyset, x')$. Hence $x = \tau X$, $x' = Y$ and $R = \emptyset$. By [TA1], $\nu \sqcup \eta \in [\![g]\!]$, 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 8.9 to obtain:

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

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

- [TA2].

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

  By Definition 8.12, a synchronization happens only if an internal-external choice synchronizes. Hence, $x = [\texttt{!a}\{g, R\}]\, X$ and $y = Y$ for some $Y \triangleq \texttt{?a}\{f, S\}.Y' + p$.

By Definition 8.12, the last part of the mapping for an internal choice is such as $Pfx(x, \texttt{!a}, R, Idle(X'))$. By definition of the Pfx pattern and by Lemma D.4, there exists an edge $(x, \texttt{!a}, \texttt{true}, R, X') \in E_1$ and $I_1(X') = \texttt{true}$. By Definition 8.12, the mapping for an external choice is: $[\![Y \triangleq \texttt{?a}\{f, T\}.\, Y' + q]\!] = Br(Y, \texttt{true}, \{(\texttt{?a}, f, T, Idle(Y'))\} \cup S)$ for some S. By definition of the Br 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 $(\nu \sqcup \eta) \in [\![f]\!]$, we derive $\eta \in [\![f]\!]$. Hence we can apply [DE-?][DE-!] and [DES-τ] to obtain:

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

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

- [TA3].

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

By [TA3], time can pass in a network only if all the locations of the current state are not urgent. According with Definition 8.12, this happens for: the second location in the mapping of an internal choice or the first location in the mapping of an external choice or a success location. Hence, we have several sub-cases:

(i) Assume that x derives from an internal choice and y from an external choice. By Definition 8.12 and Lemma D.4, x must be of the form $x = I_1(\tau X)$ for some $X \triangleq p \in D$ with $p = \oplus \ldots$, obtaining $I_1(\tau X) = \texttt{rdy}(p)$ and $\tau X \notin Loc_1^y$. By Definition 8.12 and Lemma D.4, y must be of the form $y = Y$ for some $Y \triangleq q \in D$ with $q = + \ldots$, obtaining $I_2(Y) = \texttt{rdy}(q) = \texttt{true}$ 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], $((\nu \sqcup \eta) + \delta) \in [\![I_1(\tau X)]\!] = \texttt{rdy}(p)$. Since $\texttt{ck}(D') \cap \texttt{ck}(D'') = \emptyset$, we derive $\nu + \delta \in \texttt{rdy}(p)$. Hence, we have:

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

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.

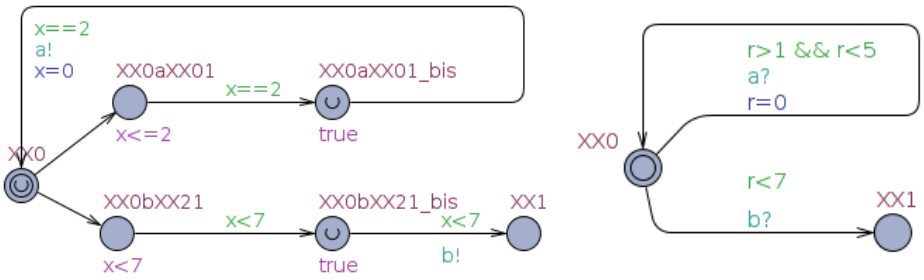

Figure 11: *Uppaal* mapping of the TSTs in Example 8.13.

D.1. **Implementation details in *Uppaal*.** *Uppaal* does not allow disjunctions in guards and invariants, because of efficiency issues. However, with little adjustments it is possible to recover disjunctions. Disjunctions in guards on edges can be achieved by creating several copies of the edge, one for each disjunct. E.g., the encoding of $!\mathtt{a}\{t < 4 \vee t > 10\}$ may give an TA with edges $(l_0, !\mathtt{a}, t < 4, \emptyset, l_1)$ and $(l_0, !\mathtt{a}, t > 10, \emptyset, l_1)$ for some l_0 and l_1.

An invariant with disjunctions invariant can be split into its disjuncts, and the location can be multiplied accordingly. In Figure 11 we see the actual implementation of Example 8.13: location names are different for what seen in the theory, since they are immaterial when checking deadlock-freedom. The invariant of location τX is $t < 7 \vee c \leq 2$, so we split it in two clauses $t < 7$ and $c \leq 2$, and each clause is assigned to a new location ($XX0aXX01$ and $XX0bXX21$). The resulting network is no longer bisimilar to the previous one: e.g., in the original network time can delay until one of the two guards is still true, while in the one in Figure 11 it can delay, non-deterministically, sometimes until 2 and sometimes until 7 time unit. Nevertheless, both networks are equivalent respect to deadlock-freedom.

Finally, since *Uppaal* supports *urgent* locations, no other adjustments are needed.

Appendix E. Proofs for Section 9

Proof of Lemma 9.7. Let us consider the LTS $\rightarrow$ of Figure 1 and the set of success states $S_1 = \{(\mathbf{1}, \nu) \mid (q, \eta) \mid q \text{ TST}, \nu, \eta \in \mathbb{V}\}$. Also, consider the LTS $\twoheadrightarrow_M$ of Figure 10 and the set of success states $S_2 = \{(\mathbf{1}, [], \nu) \parallel (q, [], \eta) \mid q \text{ TST}, \nu, \eta \in \mathbb{V}\}$. Let $\mathcal{R}$ be the relation:

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

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 show that s_2 turn-simulates s_1 via $\mathcal{R}$, and in **Part B** 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 (1), (2), and (3) of Definition 9.6 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 (1) of Definition 9.6, we consider the possible moves of s_1:

– [S-⊕]. We have:

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

where the premise requires $p = \mathtt{!a}\{g, R\}.p' \oplus p''$ and $\nu \in [\![g]\!]$. Hence, by rule [M-⊕] 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 (2) of Definition 9.6, we consider the possible moves of s_2:

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

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

So, by rules [⊕] and [S-⊕] we have:

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

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 $\nu + \delta \in \mathtt{rdy}(p)$, $\eta + \delta \in \mathtt{rdy}(q)$, 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 (3) of Definition 9.6, 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 $s_1 = ([\mathtt{!a}\{g, R\}]\, p, \nu) \mid (q, \eta)$ and $s_2 = (p, [\mathtt{!a}], R[\nu]) \parallel (q, [], \eta)$ with $\nu \in [\![g]\!]$.

(a) To prove item (1) of Definition 9.6, we consider the possible moves of s_1:

– [S-⊕]. We have:

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

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

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

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

– [S-τ]. We have:

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

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

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

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 (2) of Definition 9.6, 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:

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

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 (3) of Definition 9.6, 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 (1) of Definition 9.6, 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:

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

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 (2) of Definition 9.6, we consider the possible moves of s_1:
– [S-⊕]. 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 (2) of Definition 9.6, 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 (1) of Definition 9.6, we consider the possible moves of s_2:
– [M-⊕]. 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 (2) of Definition 9.6, we consider the possible moves of s_1:
– [S-⊕]. 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 3, 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.

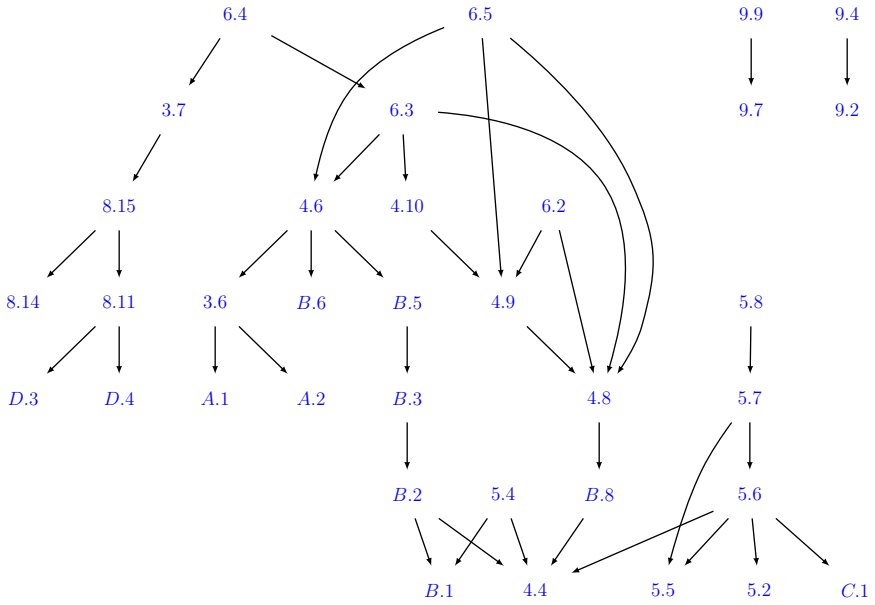

Figure 12: Dependencies among the proofs.

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

The diagram in Figure 12 illustrates the dependencies among the proofs.